TAMPERE UNIVERSITY OF TECHNOLOGY

JOHANNA KALLI

INTEGRATING A DISTRIBUTION MANAGEMENT SYSTEM AND A WORK MANAGEMENT SYSTEM USING STANDARD INTERFACES

Master of Science Thesis

Examiners: Professor Hannu Koivisto and Professor Sami Repo
Examiners and topic approved in the Computing and Electrical Engineering Faculty Council meeting on 8th of November 2013

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Electrical Engineering
KALLI, JOHANNA: Integrating a Distribution Management System and a Work Management System using Standard Interfaces
Master of Science Thesis, 80 pages, 2 Appendix pages
MARCH 2014
Major: Electrical Energy
Examiners: Professor Hannu Koivisto and Professor Sami Repo
Keywords: Common Information Model, Distribution Management System, Work Management System

Distribution Network Operators (DNO) are encouraged to improve their operation to prevent outages and to decrease overall outage times in order to improve the reliability of electricity distribution. The integration of a Distribution Management System (DMS) and a Work Management System (WMS) enables DNOs to decrease the amount of outages and to shorter the outage times. In addition, this system integration can also be used to improve customer service during outages.

A concrete outcome of this thesis is an integration of the ABB MicroSCADA Pro DMS600 distribution management system and a work management system. Use of standard interfaces is evaluated in this information system integration. Standard interfaces are expected to ease the implementation and the maintenance of information system integration. The suitability of the Common Information Model (CIM) for the DMS-WMS integration is evaluated here.

This thesis covers the system integration process starting from capturing requirements, designing the interface according to the requirements and evaluating the implementation, maintenance and testing of the integration. Different methods and techniques are discussed. As a result, it is noted that the CIM standard cannot directly be applied to integrate the DMS and the WMS and some alterations are required. However, it was concluded that using the CIM as a basis data model for the interface eased the design and the implementation of the system integration.

TIIVISTELMÄ

TAMPERE TEKNILLINEN YLIOPISTO
Sähkötekniikan koulutusohjelma
KALLI, JOHANNA: Käytöntukijärjestelmän ja työnohjausjärjestelmän integrointi standardi rajapintoja käyttäen
Diplomityö, 80 sivua, 2 liitesivua
MAALISKUU 2014
Pääaine: Sähköenergia
Tarkastajat: professori Hannu Koivisto ja professori Sami Repo
Avainsanat: Common Information Model, Käytöntukijärjestelmä, Työnohjausjärjestelmä

Sähköverkkoyhtiöihin kohdistuvat yhä tiukemmat suositukset sähkön toimitusvarmuuden parantamiseksi ja keskeytysten ajalta maksettavat vakiokorvaukset ohjaavat sähköverkkoyhtiöitä kiinnittämään entistä enemmän huomiota verkon kunnossapitoon ja vikatilanteiden hallintaan.

Suomessa sähkönjakelussa on totuttu laajasti hyödyntämään tietojärjestelmiä tukemaan sähköverkkoliiketoiminnan eri osa-alueita. Sähkönjakeluprosessin kannalta keskeisimpiä järjestelmiä ovat käytöntuki- ja käytönvalvontajärjestelmät, jotka käyttävät toiminnoissaan muun muassa asiakas-, verkko- ja mittaustietojärjestelmien sisältämää tietoa. Järjestelmien integrointi mahdollistaa automaattisen tiedonsiirron ja varmistaa tiedon yhtenäisyyden. Perinteisesti sähköverkkoyhtiöiden käyttämät tietojärjestelmät on integroitu järjestelmien välille sovitettujen rajapintojen avulla.

Tässä työssä tarkastellaan ja toteutetaan ABB MicroSCADA Pro DMS 600 käytöntukijärjestelmän ja kolmannen osapuolen toteuttaman työnohjausjärjestelmän integrointi. Tämän järjestelmäintegraation avulla sähköyhtiön on mahdollista parantaa sähkön toimitusvarmuutta, lyhentää keskeytysaikoja ja parantaa asiakaspalvelua keskeytysten aikana. Järjestelmäintegraatiossa tutkitaan standardirajapintojen käyttöä ja tarkastellaan erityisesti IEC standardeista 61968 ja 61970 koostuvan CIM (Common Information Model) tietomallin soveltuvuutta järjestelmien integrointiin. Standardi rajapintoja käyttämällä on mahdollista helpottaa järjestelmäintegraatioiden ylläpitoa ja nopeuttaa uusien integraatioiden käyttöönottoa.

Työssä määritellään järjestelmäintegraatiossa käytettävä rajapinta mahdollisten käyttötapausten perusteella ja esitellään järjestelmäintegraatioissa yleisesti käytettyjä teknologioita ja metodeja. Lisäksi tarkastellaan erilaisten ohjelmistotyökalujen soveltuvuutta käytöntuki- ja työnohjausjärjestelmien integraation toteuttamiseen.

Lopputuloksena syntyy prosessi, jonka avulla käytöntukijärjestelmä ja työnohjausjärjestelmä ovat tämän hetkisillä tekniikoilla parhaiten integroitavissa. Työssä tehtyjen tarkastelujen pohjalta havaittiin, että CIM standardi ei yksinään riitä tietomalliksi käytöntukijärjestelmän ja työnohjausjärjestelmän väliseen tiedonsiirtoon standardin keskeneräisyyden vuoksi. CIM mallin käyttö rajapinnan perustana kuitenkin helpotti ja nopeutti järjestelmien välisen rajapinnan toteutusta ja käyttöönottoa.

PREFACE

This Master of Science Thesis was written in Tampere University of Technology and the topic for this work was provided by ABB Ltd Power Systems.

I want to thank the company for providing me this opportunity to write my thesis of such interesting topic. It has been an educating journey and I feel delighted to enter this particular field of business. Supervisor of this work from ABB Ltd was M.Sc. Ilkka Nikander who I would like to thank for guidance during this work. I also want to thank the gentlemen in the corner room for their advices and the kind atmosphere.

Examiners of this thesis were Professor Sami Repo from the Department of Electrical Engineering and Professor Hannu Koivisto from the Department of Automation Science and Engineering. I want to thank them both for their academic advices and guidance during this work. The department of electrical engineering I want to thank for encouraging atmosphere and interesting lectures during my studies. My friends I want to thank for all great memories gained together during our studies.

To my family, I want to express my gratitude for your encouragement in my studies and standing by me in life. And to my husband, I am so grateful for your patience and support.

Tampere, February 10th 2014

Johanna Kalli

CONTENTS

TERMS AND DEFINITIONS

Abbreviations

AM/FM/GIS	Automated Mapping/Facilities Management/Geographical Information System
AMR	Automatic Meter Reading
API	Application Programming Interface
BIS	Business Information System
CIA	Confidentiality, Integrity and Availability
CIM	Common Information Model
CIS	Customer Information System
CLEEN	Cluster for Energy and Environment
CME	Common Message Envelope
COM	Component Object Model
DMS	Distribution Management System
DNO	Distribution Network Operator
DSO	Distribution System Operator
EA	Enterprise Architect
EMV	Energy Market Authority
EPRI	Electrical Power Research Institute
ET	Finnish Energy Industries
HTTP	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
JMS	Java Message Service
LV	Low Voltage
MDMS	Meter Data Management System
MV	Medium Voltage
MRS	Meter Reading System
NE	Network Editor
.NET	Microsoft's software framework
NIS	Network Information System
NRECA	National Rural Electric Cooperative Association
SCADA	Supervisory Control and Data Acquisition System
SGEM	Smart Grids and Energy Markets
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol or Service-Oriented Architecture Protocol
UML	Unified Modeling Language
UPS	Uninterruptible Power Supply
W3C	World Wide Web Consortium

WS	Workstation
WSDL	Web Services Description Language
WMS	Work Manage System
XML	Extensible Markup Language
XSD	XML Schema

1 INTRODUCTION

Society's dependency on electricity is continuously increasing, which emphasizes the importance of reliable electricity distribution. Electricity distribution ensures delivery of electricity from high voltage transmission system to the end users. A distribution system typically consists of different types of devices operated in an extensive area. This fact significantly complicates controlling, monitoring and analyzing of the system and makes manual procedures infeasible. Hence, different kinds of information systems are widely adapted by Distribution Network Operators (DNO) to ease the electricity distribution process.

In 2011, distribution networks in Finland were damaged in a wide area by winter storms causing long outages to customers. Outages lasted over two weeks at the worst. Major disturbances and the high level of dependency on electricity initiated the Finnish Ministry of Employment and the Economy to consider new criteria for the quality of electricity supply. The new Electricity Market Act came into force in September 2013 and it enjoins that a failure of the network cannot cause an outage over 6 hours in urban areas and an outage over 36 hours in other areas [1]. Also the standard compensation payable to customers for outages was increased. In order to correspond to the demands of the new law and to prevent and to reduce interruptions in electricity supply, DNOs need to further improve their operation.

The developing business environment sets new requirements for the system environment that consists of separate information systems used in a DNO to safely and efficiently monitor and control the electricity distribution process. One of the main information systems used is a Distribution Management System (DMS), which is a supporting tool for the control center personnel to monitor and control the network state. In order to be able to answer to the needs DNOs have at the moment, system vendors have to develop their information systems to the direction of the current market situation. The improvement of outage management becomes a current issue as the new law forces DNOs to invest in reducing outages. The DMS is the information system handling fault situations and therefore a natural target for improving outage management. Integration of the DMS with a Work Management System (WMS) improves fault control and hence can help to reduce the number of outages and outage times.

In past years outsourcing maintenance and installation work in electrical network has become more common among DNOs [2]. Variable resource requirements during normal operation in comparison to emergency situations make outsourcing beneficial especially for small DNOs. Network installation and maintenance work share the same resources, but they are very much periodical and seasonal work, which again sup-

ports outsourcing these services. Outsourcing lets the DNO to focus on its main business when supportive business functions are handled by specialized service providers [3]. Information systems and data storages currently used in DNOs have originally been designed to an environment, where DNOs provide these services themselves. A study carried out in 2009 by Aminoff et al to Finnish electricity network operators found out that the main disadvantage of outsourcing services considering network maintenance was the incompatibility of information systems. In order to provide sufficient amount of information to the subcontractors, DNOs felt that they are forced to expose their information system to outside the company. Also data transfer between information systems was found difficult if systems were provided by different vendors. [4].

The DMS itself does not provide functionality to limit the amount of information exposed to a certain user group. This is a desired feature when a DNO is cooperating with subcontractors. The integration of the DMS with the WMS enables the DNO to control the amount of information shared with outsiders. This system integration enables DNOs to decide themselves if they rather share the information with their subcontractors through the DMS or if they want to limit the amount of information shared outside the company. This system integration benefits also those DNOs who have not outsourced their network maintenance work, because with this system integration they can better provide and obtain information from their own work crews operating in the field.

In a study carried out in 2005 to Finnish electrical utilities the lack of standard interfaces was stated as the most significant problem in system integration [5]. The lack of standard interfaces has led to use of utility specific interfaces, which have proven to be expensive to maintain and updates to them are hard to implement [6]. The use of standards in information system integration generalizes the interface and thus can reduce DNOs' dependency on the system vendor and eases maintenance process.

The objective of this thesis is to study the integration of the DMS and the WMS using standard interfaces. This study also provides general knowledge about information system implementation using standard interfaces. Practical implementation of this thesis is done in integrating the MicroSCADA Pro DMS600 by ABB Ltd and a WMS using standard interfaces, the emphasis on the use of a Common Information Model (CIM). The DMS600 has been integrated with a WMS before, but the technology of this integration is outdated and requires updating. Also the functional requirements of this previous system integration have changed and therefore functional requirements are studied here as well.

The theory part of this thesis consists of a literature study of information systems used in electricity distribution introduced in Chapter 2, laying emphasis on the distribution management system and the work management system. Then integration technologies are introduced in Chapter 3 and standards in Chapter 4. Requirements for the system integration are discussed in Chapter 5 and use cases derived from the requirements are listed in Chapter 6. The profile constructed in this thesis to integrate the systems and the changes done to the CIM model are explained in Chapter 7. Different implementa-

tion methods to construct a service contract for the system integration are evaluated in Chapter 8. And finally, in Chapter 9 results are presented before conclusion.

2 ELECTRICITY DISTRIBUTION AUTOMATION

Electricity Distribution Automation means automatization of electricity distribution network operations by using electronic data processing. Electronic data processing requires use of multiple information systems which communicate with each other. The use of information systems increases efficiency, quality and safety of electricity distribution [7].

Distribution automation consists of the primary process and the secondary process. The primary process consists of all devices used to build the network. Lines, cables, transformers, capacitors and switching gears are examples of devices belonging to the primary process. Secondary process of distribution automation consists of information systems, sensors, relays and data transmission systems. These are used to monitor and control the electricity distribution process. [8]

Distribution automation can further be divided into five levels, which are company, control center, substation, network and customer automation [7]. Each automation level has its own main functions which are presented in Figure 1. The pyramidal shape illustrates the number of devices utilized in each level of automation. In company level few information systems are utilized in comparison to customer level where the volume of devices is significantly larger because all the customer sites have an energy meter.

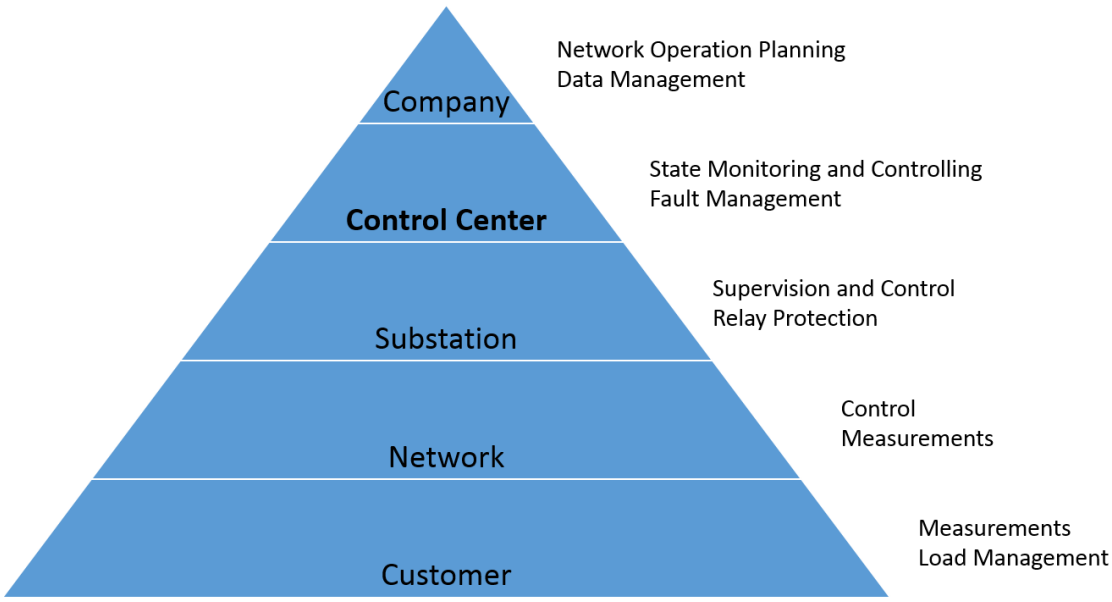


Figure 1 Levels of Distribution Automation modified from [7] and [9].

Central functions of company level automation are planning of distribution operation, management of data stored in the information systems and energy management based on meter readings. Control center automation focuses on monitoring and controlling network state and outage management. Substation automation manages protection relays, voltage and current metering and controls switching devices. Network automation is responsible for remotely controllable disconnectors, voltage and current metering and transmitting the information from fault indicators. Functions of customer automation are smart metering, load control and demand response. [7]

All these automation levels presented are necessary for electricity distribution process. The context of this thesis is focused on the control center automation and more specifically on the information systems used in control centers.

Information system can be defined as a combination of devices, software, data bases and interconnection between these all. Also the user of the information system can be considered as part of the system. Information system collects, processes, analyzes, stores and transmits information. In electricity distribution information systems are necessary for safe and efficient network operation. Categorization of information systems' functions can be challenging, because all vendors have their own visions of the purpose of their system and also the utilization of the same system may vary between DSOs. Nevertheless, the most fundamental functions of the main information systems used in electricity distribution process are presented in Figure 2 and explained in the following subsections. [7]

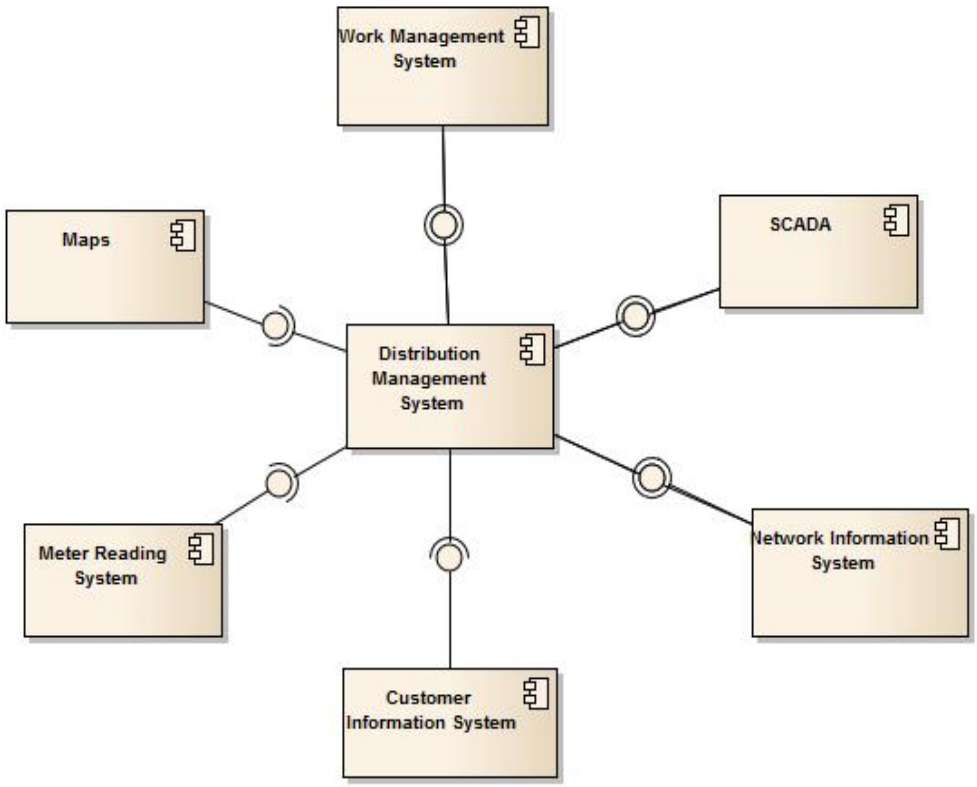


Figure 2 The main information systems used in electricity distribution

Figure 2 presents one possible system environment that could be used, but other combinations of information systems can be equally valid. DSOs may also use other information systems for business processes, such as Business Information System (BIS) or Enterprise Resource Planning (ERP), but they have been defined to be out of the scope of this thesis.

2.1 Distribution Management System

The distribution management system is a supporting tool for control center operators for real-time management and control of electricity distribution process. The DMS requires data from other information systems for its applications and that is why it has to offer several different interfaces. The main sources of information are the SCADA and the NIS. By combining static network data from the NIS and measurements and component status from the SCADA, the DMS forms a dynamic model of the network. This helps the control center to maintain the network and guides them to operate safely and efficiently.

The structure of the DMS can be presented with five layers as shown in Figure 3 below [10]. The top layer, user interface, is the layer that is visible to the user. When descending the layers down, the visibility to the user decreases.

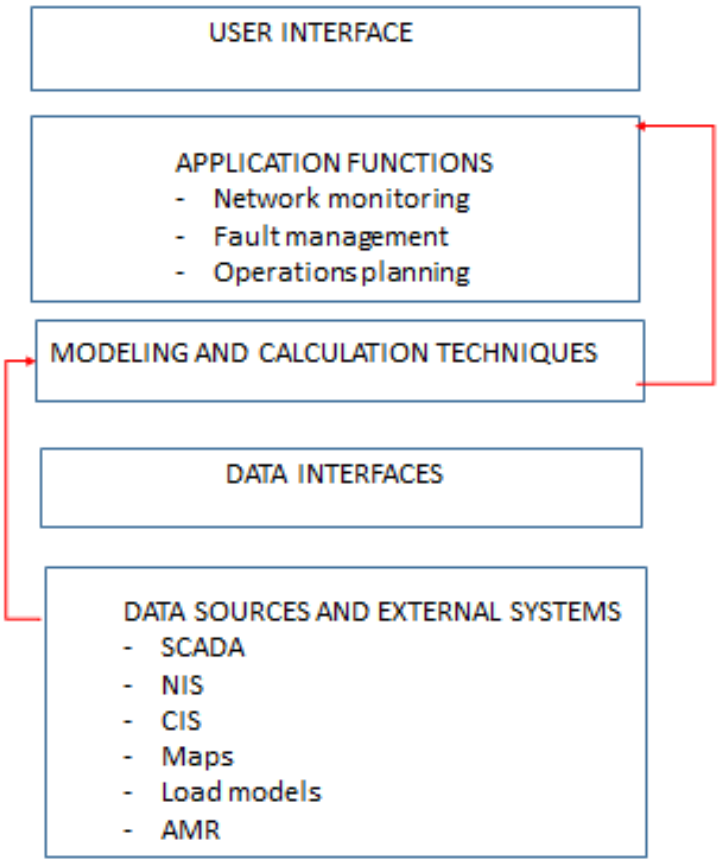


Figure 3 Structure of the DMS modified from [7]

The user interface layer describes how the user communicates with the system. The DMS provides a graphical user interface with a geographically formed network image of medium voltage network. Background map is shown depending on the zooming-scale and also low voltage network can be downloaded when needed. Different colors in the network image separate feeders from each other and line sections without electricity are drawn in white. In addition to feeder information, the DMS provides information about voltage and load levels and protection state in the network. This helps the user to visualize the structure and the state of the network quickly. Additional component information is accessible by selecting the component in the network image. [10]

The user interface of the ABB MicroSCADA Pro DMS600 is illustrated in Figure 4.

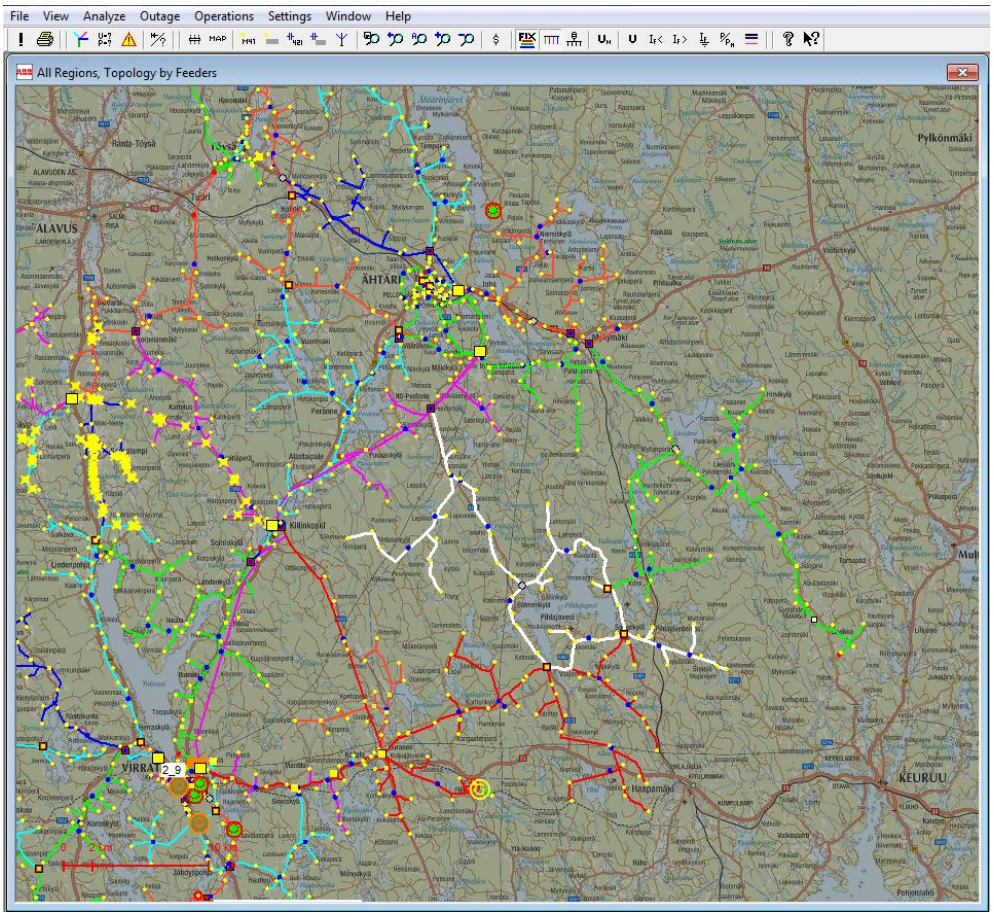


Figure 4 ABB MicroSCADA Pro DMS600 User Interface

Applications of the DMS can be divided into three main categories, which are network state monitoring, outage management and operations planning. Network state monitoring contains applications for topology supervision, field crew management, electrical state estimation and simulations. The main applications of outage management are event analysis, fault location, isolation and restoration planning, outage reporting and customer service. Operation planning consists of applications that handle switching planning, optimizing voltages and network operation. Data for all these applications comes from the modeling and calculation techniques layer, which requires data from external systems. For example, topology supervision application requires a network model, which is generated from the data the NIS provides to the DMS. [7]

Sometimes the different applications of the DMS presented here are divided into separate information systems.

The ABB MicroSCADA Pro DMS 600 consists of applications called Network Editor and Workstation. The Network Editor (NE) is a tool for editing the network data and the Workstation (WS) is used in the control center for switching state control, outage management, switching plan management and outage reporting. In order to maintain the real-time state in all client applications, a DMSSocket message is sent when

changes are made to database. Based on the socket message other client applications know that changes have occurred in the database and can refresh their data. [6]

2.2 Work Management System

The Work Management System (WMS) is an information system to collect and manage work assignments. The WMS can also be used to manage human, time and asset resources in order to complete work assignments cost-efficiently. Work assignments can vary from acute reparation and maintenance work to period based maintenance and supervision. The criticality level of the work defines the work order. From the point of view of electricity distribution operation, the WMS is a support system for other information systems used in electricity distribution and the role of the WMS system varies with network operators. The WMS itself could be used in any environment requiring tools for work management.

In electricity distribution environment, the WMS receives work assignments from the DMS. The DMS detects faults, handles maintenance plans and condition reports and reports them to the WMS. The WMS processes the information further, creates work tasks and organizes the work load. Hence, the WMS is able to provide more accurate information about outages and network condition. That information can be sent back to the DMS and used as a basis in informing and serving customers. Figure 5 presents information flow for the WMS. The WMS communicates with the DMS to receive work assignments and reports back to the DMS when more information is available. The WMS divides the assignment into work tasks and assigns a work group for each task. Work groups can update information concerning their task with a mobile application at the work site.

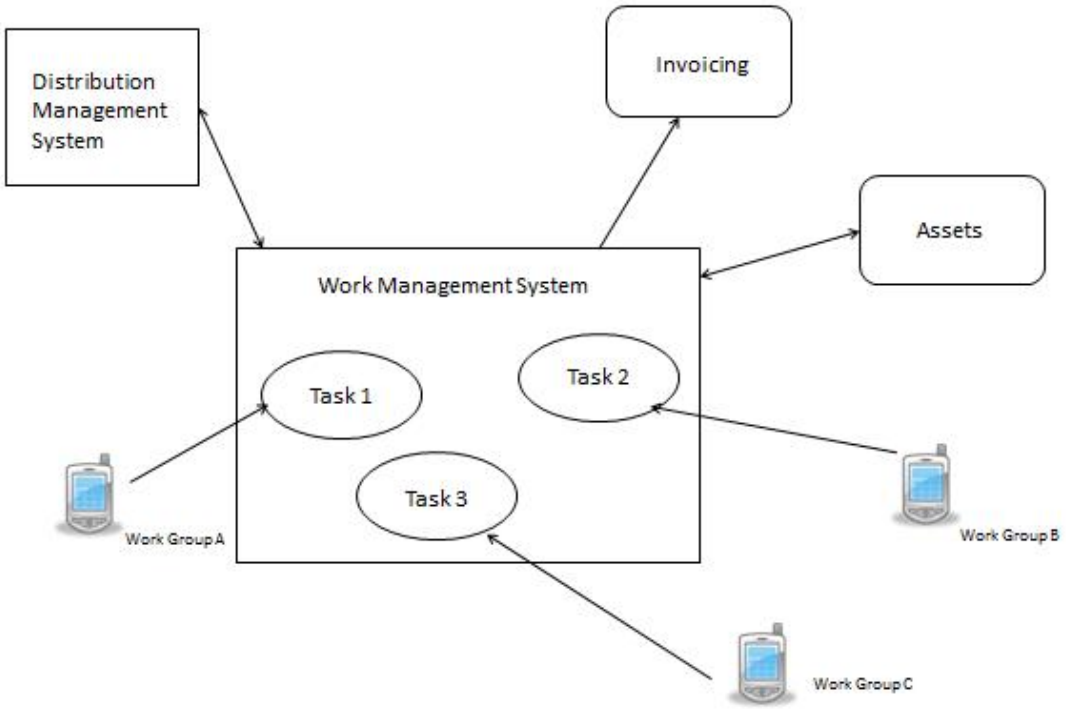


Figure 5 Description of a Work Management System

Depending on the network operator, the WMS can either be operated by the network operator or by one or more subcontractors of the network operator. In either case, the WMS enables the network operator to categorize operations and to control the amount of information exposed to outside the company.

In addition to the DMS integration, the WMS can be integrated with other information systems as well. The WMS can provide information for invoicing and assets control.

2.3 Other Systems

In addition to the DMS and the WMS other constitutive information systems used in electricity distribution are Supervisory Control and Data Acquisition System (SCADA), Network Information System (NIS), Customer Information System (CIS), Meter Reading System (MRS) and map resources. Common to all these information systems is the requirement to be able to function in exceptional conditions such as major disturbances. For that reason back up feeding systems and other precautionary measures have to be applied with these systems.

2.3.1 Supervisory Control and Data Acquisition System

The main objectives of the Supervisory Control and Data Acquisition System (SCADA) are management of event data, management of switching state in the network, remote controlling and measuring and reporting. The SCADA acquires measurement and event data from several sources and thus is able to provide real-time information about the network. Manually operated devices' state values are fed into the SCADA manually, but remotely controlled devices update their states and measurements automatically. The network information is saved into a process database, which provides real-time information for user interface and further processes. As the network information keeps changing, only the most frequent data is saved to the process database. The process database has high performance demands and therefore the capacity has to be limited. Another database, which does not have to process data in real-time, can be used for further data storage for historical analysis. The SCADA consists of a master system and remote terminal units (RTU) located at substations. In order to achieve high dependability, the whole system can be duplicated and power supply ensured by Uninterruptible Power Supply (UPS) devices. Thus failure in the network does not affect the operation of the system. [7]

2.3.2 Network Information System

Network Information System (NIS) upholds information about network assets, such as network components, their location, technical and economical characteristics and their interrelations. The main objectives of this information system are documentation of the network, network calculations, network planning and reporting. The NIS is also known as AM/FM/GIS (Automated Mapping/Facilities Management/ Geographical Information System). The main components of the NIS are network database, database management system and applications using the database information. From the user point of view the most essential applications are related to network maintenance, planning and calculation. The NIS used by DNOs is an extensive multipurpose system. It is not only used as data storage and documenting system, but also as a planning instrument. Depending on the system vendor, features of the NIS may vary from basic network documentation to comprehensive assets management. The NIS often has an interface with a DMS, a SCADA, a CIS, map resources and different business management systems. It is one of the most important information systems for the DNO as it forms the base for assets management and it provides the network topology model used in other information systems. [7]

2.3.3 Customer Information System

Customer Information System (CIS) maintains general customer information such as customer connection point information, customer group information, energy consumption information and load estimation data. From the business point of view the CIS can be considered as one of the most important information systems, because it provides information about invoicing. Before, the CIS was focused mainly on customer invoicing, but has lately developed into a system containing various applications for customer management. Also a term Customer Relationship Management (CRM) can be applied for the same system. In addition to invoicing information, the CIS is used for contract management, marketing and guiding and serving customers. Energy consumption data and customer group data provided by the CIS can further be used for network load estimation. However, the generalization of AMR meters has shifted this responsibility towards meter data management systems (MDMS). [9]

2.3.4 Meter Data Management System

Meter Data management Systems (MDMS) have become more common since AMR meters at customer sites outgrew in number and the amount of data increased exponentially. The CIS was not designed to manage the amount of measurement data the generalization of AMR meters brought and a new information system was needed for collecting, storing, analyzing and managing meter information. The MDMS consists of database storing the measurement data obtained from the Meter Reading System (MRS), tools used for data processing and interfaces with other systems such the CIS and AMR meters for obtaining required information for processing. Meter readings are used for customer invoicing, imbalance settlement, load prediction and network planning. Instead of the MDMS, the meter assets management, collection of meter readings and meter reading management can be managed with separate information systems. [3]

2.3.5 Maps

The DMS and the NIS use maps to visualize of the network topology in reality. Maps used in several information systems' applications are provided by the National Land Survey of Finland or other similar sources.

3 INTEGRATION OF INFORMATION SYSTEMS

Designing a single information system that would support all aspects of the business is nearly impossible. Enterprises also prefer combining different information systems in order to achieve the combination of applications that best correspond to their needs. This approach brings flexibility to the enterprise and lets vendors to offer applications specialized in a certain area of business. [11]

Increasing complexity of information systems requires combining data from different sources and in order to do that system integration is required. System integration helps to reduce data redundancy, as the data from one source can be reused and all systems do not have to maintain their own copy of the same data. Also the inconsistency of the data can be minimized when the data is maintained only in one system. [12]

Information system integration may seem simpler in theory than what it is in practice, because several factors have an effect on system integration. Evolving business environment sets requirements to information system, because they are an important part of the overall business architecture. Ideally, information systems would be able to respond to the changing needs of the business and system integrations would be so flexible that unpredictable information exchange and data analysis could be implemented afterwards. Information systems are very heterogeneous, using different platforms, data models and programming languages, which makes managing of the overall system architecture very challenging. [13]

Information systems in electrical utilities are typically investments from different decades and based on different technologies [2]. Therefore, systems environments are specific to each utility and requirements for system integration are not identical. Purchasing systems from different vendors can create an incompatibility issue, because all vendors have their own data models that are not compatible with each other. To avoid the incompatibility issue, the utility can try to purchase all information systems and integration services from one vendor. This simplifies the integration and the maintenance process as the internal data model is the same for all the systems. However, it creates a huge risk for the utility's business as the utility becomes very dependent on one vendor. It is also rare that one vendor can provide all services for the utility and for reasonable price. Eventually this arrangement can lead to a situation where utility's chances to tender new projects in future are nonexistent. On the other hand tendering all the projects and using multiple vendors diminishes risks for business, but can lead to uncontrollable system architecture. Use of multiple vendors requires implementing incompatible data models together and therefore causes extra work in the integration process. It is clear that vendors are accountable for their own systems, but interfaces be-

tween systems by different vendors are not so unambiguous. In a long run, maintenance can be expensive and many sources for accountability can lead to misunderstandings in maintenance responsibilities.

System integration has implications on the overall business both positively and negatively. Integration of critical business solutions is vital source of information, but at the same time the business becomes dependent on that integration solution and system failure could result in remarkable economic losses. That is why system integrations are so essential and have to be planned carefully. [11]

Over time different integration architectures have evolved to solve issues in system integration. The following subsections present different integration architectures and their benefits and later technologies used for system integration in this thesis are reviewed.

3.1 Point-to-point Integration

A point-to-point integration means directly connecting two systems together so that they can communicate and exchange information with each other. Point-to-point integration can be seen as the simplest architecture in system integration and it is successfully applied in many system integrations. Benefits of point-to-point integration are that it is easy and straightforward to implement. In Figure 6 below point-to-point integration is applied to simple system integration on the left and to more complicated system environment on the right. Each ball represents an information system and red lines connections between integrated systems.

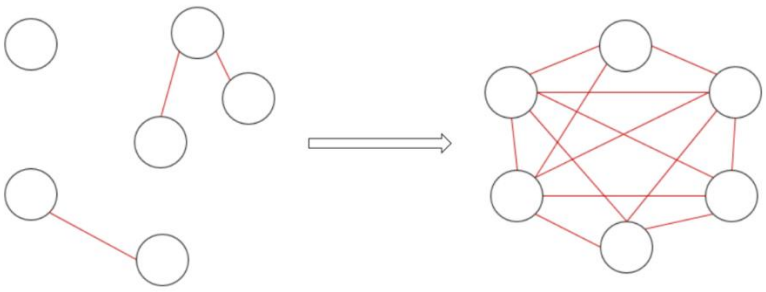


Figure 6 Evolution of point-to-point integration

If all the systems are integrated with each other, the amount of connections between systems increase in square to the number of systems. Thus, if there are n systems to be integrated with each other, the number of connections is equal to $n(n-1)/2$. When the number of systems increase, updating and maintaining the overall system becomes hard and eventually impossible. Therefore it is justified to say that this archtecture is not ablicable in complicated system integrations due to the scalability issue. [14]

When the overall system architecture is beginning to look like the figure on the right in Figure 6 it is necessary to renew the point-to-point architecture to more scalable solution. However, in simple systems point-to-point architecture is preferred.

3.2 Enterprise Application Integration

Enterprise Application Integration (EAI) means a collection of procedures, tools and services to enable communication between applications. Applications can be located within an enterprise or they can be distributed.

Figure 7 presents system integration with the EAI architecture. Each ball represents an information system and the EAI in the middle enables connections between these systems. Now the required amount of connections equals the number of systems, which makes this architecture more scalable.

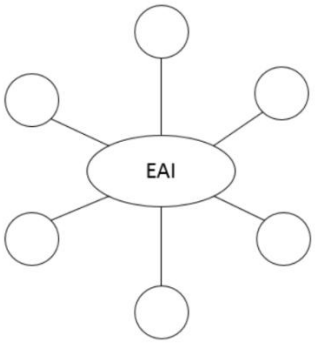


Figure 7 EAI Architecture

The EAI reduces the number of connections and thus adding a new system is faster than in point-to-point integration. In addition, central monitoring and controlling the traffic is possible, which would be hard to implement in point-to-point integration.

A hub/spoke and a bus are two different architectural styles defined for an EAI implementation. Both allow service development and thus implementation of Service Oriented Architecture (SOA) which is discussed later in section 3.4.1. The central idea of the EAI is that the hub or the bus in the middle handles all messages coming from each system and transfers them to right location in required format. The hub/bus structure can form a bottleneck in transforming messages as all the messages are transferred through it. This also increases and complicates the message path. [12]

The EAI provides more scalable solution in comparison to the point-to-point integration, but as the complexity of the overall system increases, the maintenance becomes challenging also with this implementation. EAI solutions are often criticized for their complicatedness, because the hub or the bus has to know everything and transfer all messages, which leads to high maintenance costs. Some well-known and widely used

EAI products include WebShpere by IBM, BizTalk by Microsoft and Netweaver PI by SAP [15].

3.3 Enterprise Service Bus

Enterprise Service Bus (ESB) also provides capability for implementing SOA. The ESB is a developed version of integration middleware from the Enterprise Application Integration (EAI) technology and web services. The main differences between the ESB and the EAI solution are the distributed nature of the ESB and that the ESB is more dynamic and based on web-technologies, such as the XML.

The ESB acts as a message carrier and breaks messages between applications. This reduces point-to-point connections between applications. Services that implement the integration procedures, such as data configuration and routing, are attached to the service bus like any other service [2]. Separating the application and the integration logic significantly reduces the complicatedness of the system. The bus does not have to know everything and do everything when services can handle part of that work. New services can be added as a demand for them occurs without an effect on existing applications.

Figure 8 below presents an enterprise service bus implementation. Service providers and consumers are connected through the bus, which enables controlling the overall system.

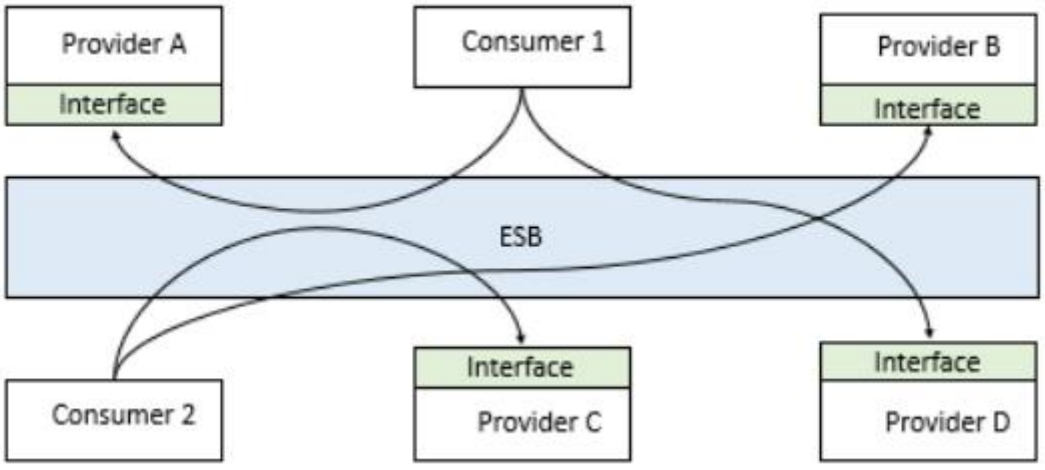


Figure 8 The Enterprise Service Bus modified from [15]

The ESB commonly provides functionalities such as adapter support, messaging layer, data mapping and routing, business process automation, security and system management and configuration environment [15]. The aim is to create a simple and easily maintainable system, which does not restrict information exchange between applications.

The ESB is found to be cheaper and easier to maintain than the EAI. In comparison to point-to-point integration, the ESB is more complicated to implement but easier to maintain and it has better scalability. In simple system environments it is not recommendable to implement the ESB, but as the size of an overall system increases, implementing the ESB is preferred.

Examples of ESB products currently in the market include IBM WebSphere ESB, Microsfot Biztalk Server and Windows Azure Service Bus.

3.4 Integration Technologies

This chapter introduces technologies used for information system integration in this thesis. Because the DMS and the WMS are designed and possibly operated by different parties, the system integration is done over the web. Therefore, systems do not have to be located in the same system environment and communication is possible between control center personnel operating the DMS and field crews operating the WMS with their mobile equipment.

Due to the heterogeneous nature of the web, communication technologies must be platform-independent [16]. Service Oriented Architecture (SOA) emphasizes platform independency and the technologies have been chosen to actualize the SOA concept. Also system environments where these two systems are running, data volumes and other nonfunctional requirements discussed earlier have an effect on technology decisions. Figure 9 presents relations between the technologies used in the implementation of the DMS-WMS integration.

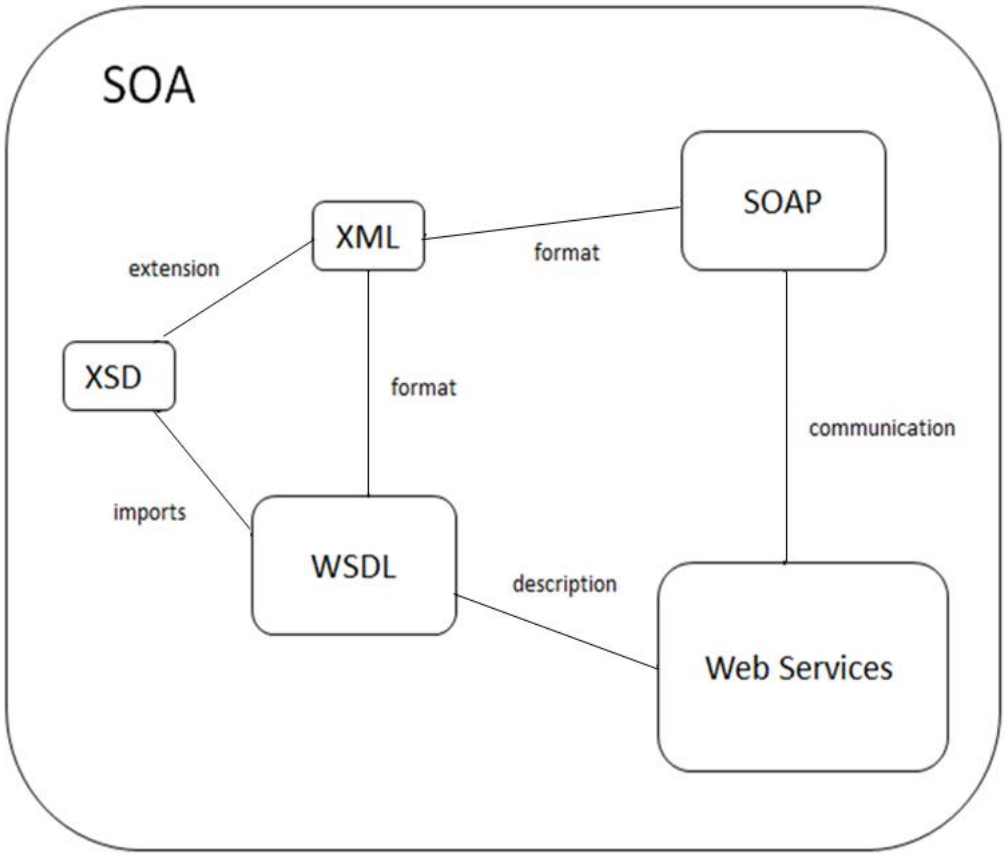


Figure 9 Relationships between used technologies modified from [17]

The SOA is a design pattern, which is implemented by using web services. Web services use SOAP to define communication between service providers and service consumers. WSDL is used to describe the service contract. The SOAP and the WSDL are XML format descriptions or extensions. The WSDL can import XSD definitions. XSD, also known as XML Schema is an extension to XML format. Each of these specifications is explained in more detail in the following sub sections.

3.4.1 Service Oriented Architecture

Service Oriented Architecture (SOA) is a software design pattern that aims to simplify software development process and increase agility and flexibility. Benefits of the SOA are platform independency, loose coupling, discoverability, and clear separation of the service provider and the consumer. The SOA discourages use of point-to-point integration and encourages loose coupling between information systems. Loose coupling means that information systems are integrated so that they are dependent on each other as little as possible. The interface of a service can be understood as a contract defining how the service can be used and what does it provide for the consumer. Once defined, services are discoverable to all applications and information systems that want to con-

sume the service. Services are used over information networks such as Internet. The platform independency ensures that the service provider and the consumer can use different system environments, and therefore they do not have to be from the same organization or a company.

The SOA is not limited to one technology or standard, but instead it can be understood as method of dividing functionalities into blocks. These blocks describe services and by combining different services together, business processes can be performed [18]. This approach allows new services to be added later to the system when a need for them appears and the whole system does not have to be planned at once. These service components can be reused for other programs which increases the efficiency of the implementation process.

The SOA provides a conceptual-level model and therefore detailed technology is needed to support it. For the data exchange some commonly agreed languages have been defined and even though the SOA is not dependent on any specific technology, the Extensible Markup Language (XML) and web services are often deployed for a SOA implementation. Web services are not inherently service oriented and nor does the SOA require use of web services. Nonetheless, currently web services offer the best option to supporting the SOA. [19]

3.4.2 Web Services

According to W3C definition web services are ‘software systems designed to support interoperable machine to machine interaction over a network’ [26]. Web Services define a set of protocols by which services can be created, published, discovered and consumed in a standard form. Web services can further be divided into communication protocols, service description and service discovery. A communication protocol used in this thesis is SOAP and service is described in a machine readable format, Web Service Description Language (WSDL). Use of the WSDL is not a SOAP requirement, but a prerequisite for automated code generation. Universal Description, Discovery and Integration (UDDI) specification is commonly used for discovering services. In the context of this thesis, service consumers know the service they are connecting with, so UDDI is not applied in this case.

Web services can be divided into two different approaches, which are document-centric approach and Remote Procedure Call (RPC) approach. The current trend favors document-centric approach over the RPC approach, because the latter does not support the SOA and loose coupling between the service provider and consumers. Also the IEC 61968-100 standard applied in this thesis focuses on document-centric web services.

The term web services can generally be applied to many different systems, but here the term is used to refer to systems that consist of clients and servers communicating with each other over a network using SOAP standard XML messages. Web services receive and send data in XML formatted documents, which are platform independent. Only the service contract is visible to service consumers, and the internal im-

plementation of the service is hidden. Technologies used to implement web services also enable web services to be located anywhere and still be accessible by service consumers. [16]

In the traditional server- client model, the web service corresponds to the server and the service consumer to the client as presented in Figure 10. The service consumer sends a request message to the service, which processes the request and responds back to the client by sending a response message.

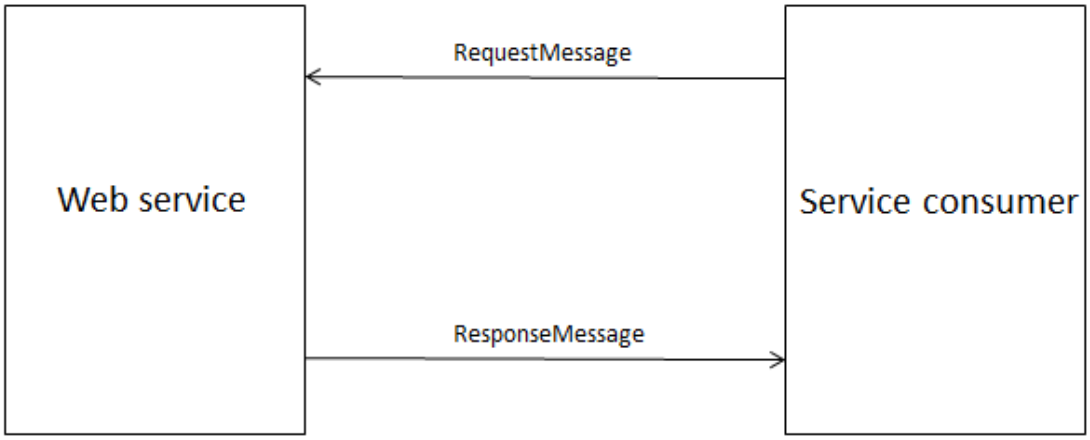


Figure 10 Web services presented as traditional server-client implementation

Even though the web services are widely used and provide an easy way to connect applications regardless of the platform, it is not a silver bullet technology that would solve all problems related to information system integration. [2]

3.4.3 Extensible Markup Language

Extensible Markup Language (XML) is a World Wide Web Consortium (W3C) standard for defining data structure. XML is stored in human readable text format, which can also be parsed by computer. Simplicity, generality and usability have made XML the most common data exchange format over the Internet. It is the lingua franca for information and data encoding internationally [16]. XML is an independent format, which enables software developers working with incompatible systems to exchange data between applications.

XML is a self-descriptive markup language to create structure in the presented data. The information is marked with tags that form hierarchical structure for the data. XML documents are composed of storage units called entities, which define the contents of the document [21].

Figure 11 shows an example of a valid XML file and the corresponding structure of elements, where namespace is equal to http://localhost/Example. Use of XML namespaces help to avoid name conflicts.

```
<?xml version"1.0" encoding="UTF-8" ?>
<Note xmlns:n="http://localhost/Example">
        <n:id>1234</n:id>
        <n:description>This is example.</n:description>
</Note>
```

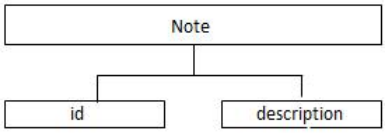


Figure 11 An example of valid XML file and corresponding structure

Element called Note contains two other elements called id and description. More elements could be added to Note or to its child elements without breaking any applications using this XML description, because XML elements are extensible.

XML can further be specified with accompanying standards such as XML Schema (XSD) for defining allowed elements in XML file, WSDL to describe web services and XLink for creating hyperlinks in XML documents. There are many more extensions to XML not listed here. An XML implementation does not have to support all available extensions.

3.4.4 Simple Object Access Protocol

Simple Object Access Protocol (SOAP) is XML based protocol designed to exchange structured information in distributed environment over Hypertext Transfer Protocol (HTTP). It is also a W3C recommendation and it has been widely adapted in application development for communication over Internet, because it provides a way to communicate between applications implemented with different technologies, programming languages and running on different operating systems. [22]

The main elements of a SOAP message are an envelope, a header and a body. The envelope is the root element of the message as it defines the XML document as a SOAP message. The header element is optional and it contains application specific metadata, which describes how the recipient should process the SOAP message. The body element contains the actual message part for the receiver. The body content is application specific information and not a part of the SOAP namespace. Other standards can be applied to extend SOAP header vocabulary and the body content. These standards are called web service activities.

A SOAP message is formed by an XML element called Envelope, which contains two child elements, the Header and the Body. Contents of the Header and the Body depend on the message. The basic structure of a SOAP message is presented in Figure 12.

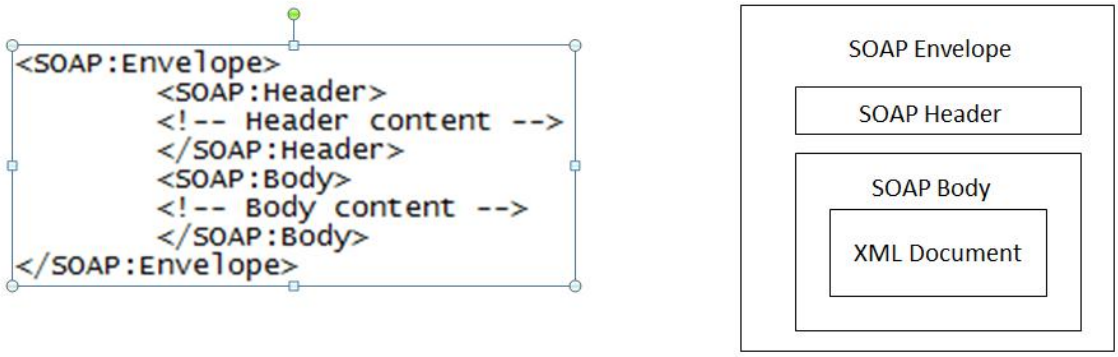


Figure 12 Structure of a SOAP message [16]

The SOAP offers basic communication method, but it does not provide information about messages, which are used in web services communication. For that purpose another XML based format, WSDL, has been developed. [16]

3.4.5 Web Service Description Language

Web Service Description Language (WSDL) describes interfaces between solutions and provides a contact point for service users to connect with the service. The contact point is also called an endpoint of the service.

A WSDL document consists of two parts, abstract definition and protocol-dependent definition. The abstract definition can further be divided into three components; the vocabulary, the message and the interaction. The vocabulary defines the data types used in the information exchange and it is often determined by external data source such as XML Schema file. The message element in turn, provides an abstract definition of data types received and sent by the service. The interaction is defined by a combination of operation and portType elements. The operation element describes a pattern for the message exchange that the web service supports and the portType element gathers those operations together that are supported by a certain endpoint. [16]

Figure 13 shows an example of a port that has one operation, which takes one type of message as an input parameter and another type of message, as an output parameter. In this example, portType called ExamplePortType has an operation called GetNote. The GetNote operation expects GetNoteRequest type of message as an input and sends a message of type GetNoteResponse as an output back to the request sender. The GetNoteRequest message has two parts, id and description. Both parts have external XSD definitions imported to the WSDL file. The id is defined as XSD integer and the description as XSD string.

```
<message name="GetNoteRequest">
        <part name="id" type="xsd:int"/>
        <part name="description" type="xsd:string"/>
</message>
<message name="GetNoteResponse">
        <part name="Note" type="fixsd:NoteType"/>
</message>

<portType name="ExamplePortType">
        <operation name="GetNote">
                <input message="tns:GetNoteRequest"/>
                <output message="tns:GetNoteResponse"/>
        </operation>
</portType>
```

Figure 13 Abstract part of WSDL

The protocol-dependent level describes what protocol to use, how to use it and where the endpoint for the service is located. The binding element of the protocol-level describes how to connect with the used protocol and the port element is used to define the endpoint, which processes requests for the service. [16]

```
<binding name="NoteSenderSoapBinding" type="tns:ExamplePortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetNote">
    <soap:operation style="document" soapAction="http://localhost/example"/>

    <input name="GetNoteRequest">
      <soap:body use="literal" namespace="http://localhost/example"
    </input>

    <output name="GetNoteResponse">
      <soap:body use="literal" namespace="http://localhost/example"
    </output>
  </operation>
</binding>

<service name="NoteSender">
  <port name="NoteSenderPort" binding="tns:NoteSenderSoapBinding">
    <soap:address location="http://localhost/NoteSender"/>
  </port>
</service>
```

Figure 14 Binding and Service of WSDL

Figure 14 shows how the binding information in presented in a WSDL document. The binding element describes how to connect with the service called NoteSender by using SOAP protocol. The previously defined operation called GetNote is defined to be document-oriented, which means that it uses XSD to describe the transmitted XML. The port element identifies the location of the endpoint. In this example, service called NoteSender is located at http://localhost/NoteSender and it is defined to use SOAP re-

quests. The number of service endpoints is not limited to one, but in this example only one endpoint has been defined.

A WSDL document is a contract between the service provider and the consumer about the content of the service and instructions how to connect with it. The document is extensible, so new endpoints can be added as the service develops without affecting the usage of existing service contracts.

Two distinctive methods exist for creating WSDL documents. First method is called bottom-up method, which means an approach where the service implementation is written first and the WSDL document generated based on the code. The other method called top down or contract-first has totally opposite approach for creating WSDL documents. According to this method, the service contract is created first and the code skeleton generated from the contract. These two different approaches are discussed in more detail in chapter 8. The IEC 61968-100 standard discusses two approaches concerning WSDL definition and web services. Those two options are generic WSDL operations and strongly typed WSDL operations. These are also covered in more detail in chapter 8 [23].

3.5 Server client communication

This chapter introduces possible techniques for transferring data between the DMS and the WMS. This system integration is required to enable two-way connection over the web. Different techniques exist to transfer information over the web and four most applicable methods are introduced in the following subsections.

Technologies to communicate over the web can be divided into push and pull techniques. Traditional web architecture is originally designed to support pull technique, where clients actively request changes from the server. No permanent connection between a client and a server is established here. Need for other interaction style has arisen, because advancing web applications require real-time event notifications. Server push technology has been developed to allow server to broadcast data when changes occur on the server side. [24]

3.5.1 Traditional server-client technique

The traditional server-client model is based on functionality that the client sends a request message to the server. Server processes the request and then returns an applicable reply message back to the client. With this method, client always initiates the communication process. Figure 15 demonstrates the traditional communication between client and server.

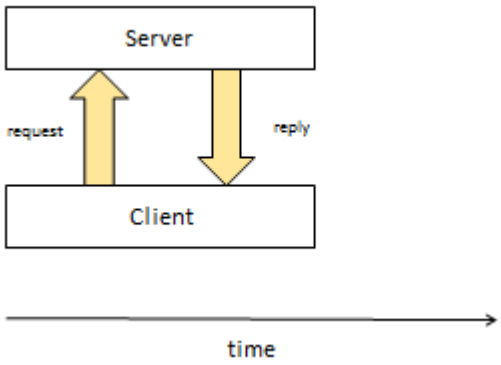


Figure 15 Traditional client-server communication method

When the server is located behind a firewall, the firewall must be opened to enable communication between the server and the client. Another disadvantage of the traditional server-client method appears when completing the process at the server requires a lot of time. This method does not guarantee that the connection stays open long enough that the client is available to receive the reply message.

3.5.2 Polling

Polling is another communication method initiated by the client. Polling means that the client actively sends requests to the server to check if the status of the server has changed. Polling interval determines how often the request is sent. If the server status has not been updated since the last request message, an empty reply message is sent back to the client. When an update occurs, reply message with message content is returned to the client. Figure 16 illustrates a client polling a server.

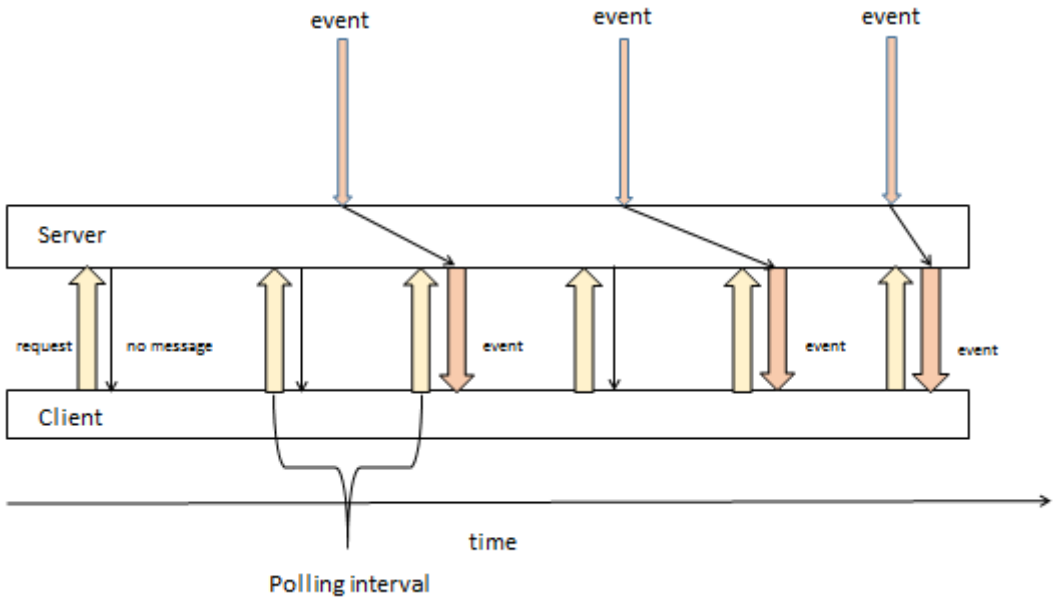


Figure 16 Polling adapted from [25]

Polling is an efficient method when client requires one notification to complete, but this method is not applicable with large systems. With polling, balancing between server load and latency is always present. Long polling interval reduces server load, but increases latency and decreases accuracy and freshness of the data. Short polling interval, in turn improves data quality but increases server load. [24]

3.5.3 Long polling

Long polling is a combination of the polling technique and traditional server-client model. The client polls the server with request messages, but at slower frequency than in traditional polling. The server receives the request from the client, but instead of immediately returning a reply message, the server holds the request open until it has some new information available. Figure 17 illustrates long polling communication.

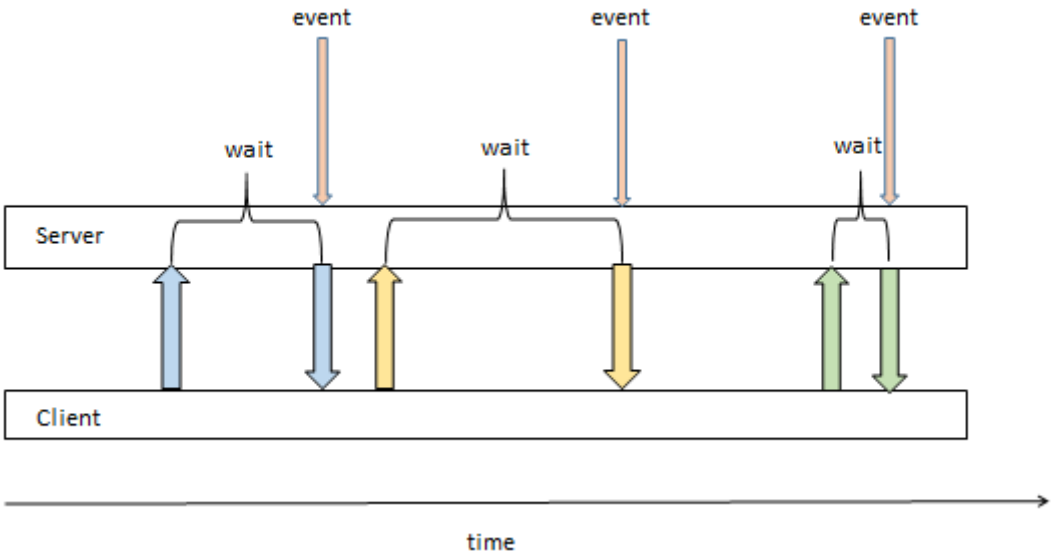


Figure 17 Long polling adapted from [25]

Long polling reminds server push techniques, but because the client first has to open the connection, long polling cannot be considered as a true server push technique.

3.5.4 Server Push

A technology where the communication is initiated by the server is called server push or broadcasting. Server push is based on publish/subscribe model, where the server publishes information to a channel and the client subscribes the same channel to receive the information. Server push technology is used, for example, in instant messaging and e-mail.

Figure 18 demonstrates how server side events trigger the server to send response messages to the client. The first arrow in the picture represents that the client has to first announce that it wants to subscribe publications from the server.

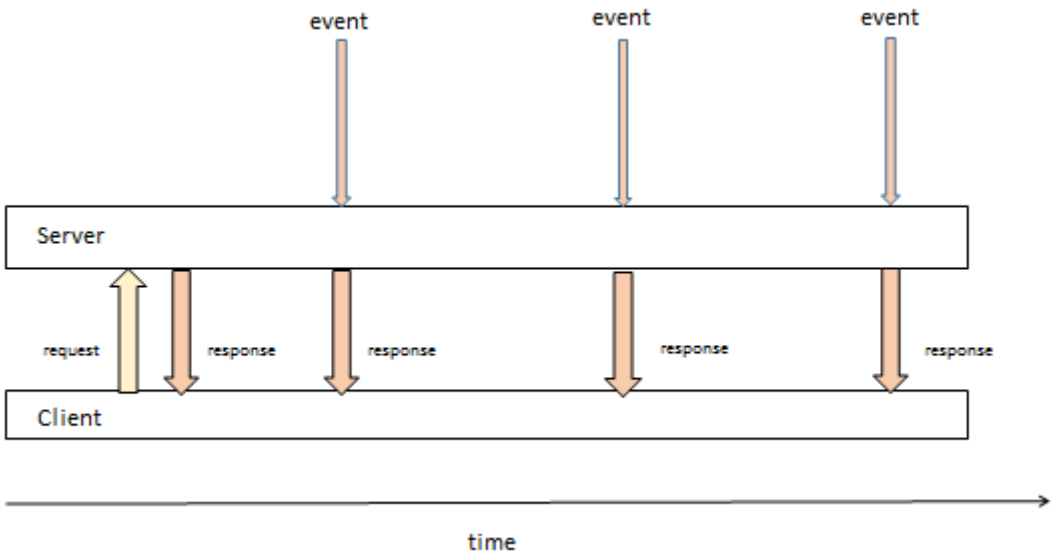


Figure 18 Server Push Technique

Server push technique can be used to reduce data access latency [26]. This approach is especially useful with ESB, because it enables the server to publish updates to the ESB that can further distribute server updates to subscribing applications.

4 STANDARDS

According to Robinson and Zhou in [27], information systems used in electrical utilities should be integrated by using canonical data model (CDM) instead of integrating systems directly with each other. Typically integration projects focus more on integration technology rather than semantics, which leads to complex and slow integrations as the data remains in numerous incongruent formats and cannot be managed systematically. The CDM offers an integration layer that enables systems using different internal data models to communicate with each other. As the number of data transformations between the systems is reduced, also the scalability of the system environment increases. [27]

This chapter discusses applicable standards for integrating the DMS and the WMS. The CIM and the MultiSpeak are the two most implemented data models in electrical utilities worldwide. These two standards have been developed independently from each other and recently there has been growing effort to harmonize the MultiSpeak with the IEC CIM standard [28]. This thesis focuses on studying the IEC Common Information Model (CIM) standard, but other possible standards are covered as well. One aim of this thesis is to evaluate the usability of the CIM standard in system integration.

4.1 Common Information Model

The Common Information Model (CIM) is an abstract information model used for power system management and information exchange. The CIM was originally developed by the Electrical Power Research Institute (EPRI) in North America in 1993 to define common definitions for power system components for Energy Management System (EMS) Application Programming Interface (API). Now the CIM consists of a series of open standards under International Electrotechnical Commission (IEC). In addition to IEC, the CIM is recognized also by EPRI and IEEE. [29]

Standards forming the CIM are the IEC 61968 System interfaces for distribution management (SIDM) and the IEC 61970 Energy management system application program interfaces (EMS-API) developed by the IEC Technical Committee 57. According to the CIM Primer by EPRI the IEC 61970-301, the IEC 61968-11 and the IEC 62325-301 CIM Market Model are collectively known as the CIM for power systems. The IEC 61970-301 describes the components of the power system and the IEC 61968-11 extends the semantic model of the network to cover aspects such as asset tracking, work scheduling and customer billing. The 62325-301 in turn covers data exchange model

between participants in electricity markets. The document states three primary uses for the CIM, which are; "facilitate the exchange of power system network data between organizations; to allow the exchange of data between applications within an organization; and to exchange market data between organizations". [29]

Because the CIM consists of series of standards covering a wide range of topics, it has grown to be a large and extensive model. The CIM is documented in Unified Modeling Language (UML), which is a graphical modeling language widely adapted in software engineering. The UML class diagram presentation of the CIM attempts to visualize the extensive model to make it more usable and easier to approach. [29]
The CIM standard is divided into two packages described in Table 1 and in standard.

Table 2 below. Some parts of the standard marked unpublished in the tables are available at the CIM user group, but officially those parts are still under development. The whole standard can be downloaded at CIM user group homepage and it can be viewed with SPARX EA Lite read only version, which does not require purchasing any license. [30].

Table 1 presents the current state of the IEC 61968 standard. The main parts of the IEC 61968 standard are general interface architecture (part 1), glossary (part 2), interface definitions for network operations (parts 3 – 10), an extension to the IEC 61970-301 standard (part 11), use case examples (part 12), description of a model exchange format (part 13), harmonization of the CIM and MultiSpeak standards (part 14) and description of implementation of CIM profiles (part 100).

Table 1 IEC 61968 System interfaces for distribution management (SIDM) [31]

Part	Description	State
IEC 61968-1	Interface architecture and general recommendations (Ed. 2.0)	2012-11
IEC 61968-2	Glossary (Ed. 2.0)	2011-03
IEC 61968-3	Interface for network operations (Ed. 1.0)	2004-03
IEC 61968-4	Interfaces for records and asset management (Ed. 1.0)	2007-07
IEC 61968-5	Operational Planning and Optimization (OP)	Unpublished
IEC 61968-6	Interfaces for maintenance and construction (Ed. 1.0)	2014-08
IEC 61968-7	Network Extension Planning (NE)	Unpublished
IEC 61968-8	Interface Standard For Customer Support (Ed. 1.0)	2014-05
IEC 61968-9	Interfaces for meter reading and control (Ed. 2.0)	2013-10
IEC 61968 - 10	Interfaces for Business functions external to distribution management	Unpublished
IEC 61968 - 11	Common information model (CIM) extensions for distribution(Ed. 2.0)	2013-03
IEC 61968 - 12	Common information model (CIM) Use Cases	Unpublished
IEC 61968 - 13	CIM RDF Model exchange format for distribution (Ed. 1.0)	2008-06
IEC 61968 - 14	CIM Harmonisation (Ed. 1.0)	2013-07
IEC 61968-100	Implementation profiles (Ed. 1.0)	2013-07

The IEC 61970 standard consists of common guidelines (part 1), glossary (part 2), description of an abstract data model (part 3), abstract interface definitions (part 4) and descriptions of techniques used to construct CIM based interfaces (part 5). Table 2 describes the different parts of the IEC 61970 standard.

Table 2 IEC 61970 Energy management system application program interfaces (EMS-API) [31]

Part	Description	State
IEC 61970-1	Guidelines and general requirements (Ed. 1.0)	2005-12
IEC 61970-2	Glossary (Ed. 1.0)	2004-07
IEC 61970 - 301	Common information model (CIM) base (Ed. 5.0)	2014-01
IEC 61970 - 302	Common information model (CIM) financial, energy scheduling and reservations (Ed. 1.0)	2012-03
IEC 61970 - 401	Component interface specification (CIS) framework (Ed. 1.0)	2005-09
IEC 61970 - 402	Common data access facility (Ed. 1.0)	2003-02
IEC 61970 - 403	Generic data access (Ed. 1.0)	2008-06
IEC 61970 - 404	High Speed data Access (HSDA) (Ed. 1.0)	2007-08
IEC 61970 - 405	Generic eventing and Subscription (GES) (Ed. 1.0)	2007-08
IEC 61970 - 407	Time Series Data Access (TSDA) (Ed. 1.0)	2007-08
IEC 61970 - 452	CIM Static transmission network model profiles (Ed. 1.0)	2013-08
IEC 61970 - 453	Diagram layout profile	2014-04
IEC 61970 - 456	Solved power system state profiles (Ed. 1.0)	2013-05
IEC 61970 - 458	Common information model (CIM) extension to generation	2014-01
IEC 61970 - 501	Common Information Model Resource Description Framework (CIM RDF) schema	2006-03
IEC 61970 - 502	Web Services Profile for 61970-4 Abstract Services	2011-05
IEC 61970 - 552	CIM XML Model Exchange Format	2013-10
IEC 61970 - 555	CIM based efficient model exchange format (CIM/E)	2013-10
IEC 61970 - 556	CIM based graphic exchange format (CIM/G)	2013-10

The IEC 61968-1 standard provides an Interface Reference Model (IRM) presented in Figure 19. The IRM identifies 14 business functions, which can be implemented by using profiles derived from classes the CIM model provides.

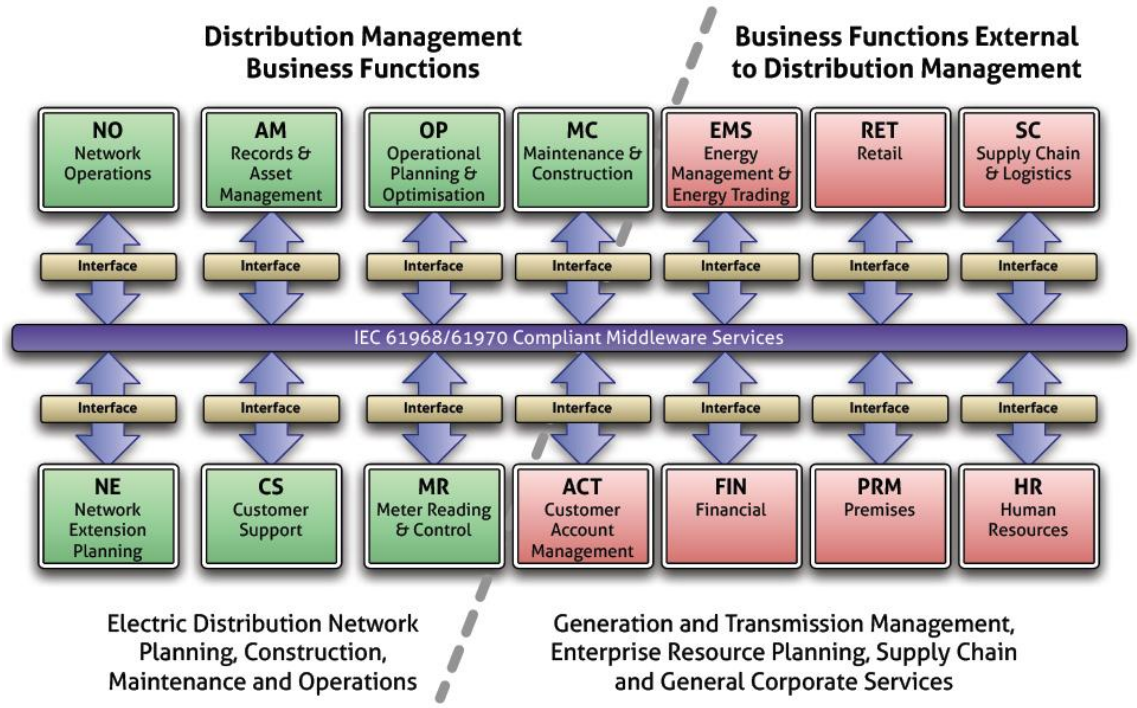


Figure 19 IEC 61968-1 Interface Reference Model [35]

The CIM defines class relations and attributes in a very general way. This leads to a possibility of very different kind of data interpretations, which are valid but incompatible at the same time. In order to solve this issue profiles are defined for each interface. Profiles consist of classes and attributes, which are defined to be either optional or mandatory information. As the profile is defined, implementation model such as XML Schema can be derived to be used for interface implementation. When the CIM model is viewed as profiles, the interface appears a lot simpler and manageable as the model of over 600 classes is restricted to contain only the classes required to present the profile. If one system has several interfaces, the profiles are derived from the same model. The data definitions can be re-used to as many profiles as necessary. [29]

Instead of constructing a profile for each interface, the CIM could in theory be implemented entirely. This is however not recommended due to the extensibility of the CIM. The interface would become very heavy and hard to understand, if the whole model was included. The IEC standards list example profiles for numerous cases including metering, work management and operations to be used in the interfaces. Because the standard is still partly incomplete as can be seen in Table 1, profiles do not exists for all interfaces defined in Figure 19. Currently, there is no profile to describe information exchange between the DMS and the WMS. A profile to satisfy the requirements of this system integration is constructed in this thesis and discussed more in chapter 5.

The CIM standard does not define implementation techniques, but it provides a standard for information exchange. The IEC 61968-1 prescribes information exchange in terms of a verb, a noun and a payload and defines a common message envelope (CME) for information exchange. The CME is introduced in Figure 20 [32].

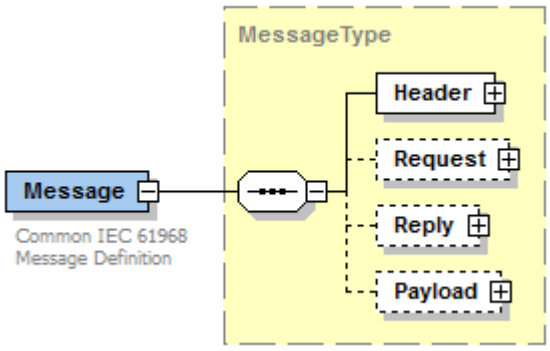


Figure 20 A Common Message Envelope [38]

The CME consists of a mandatory Header field and optional fields Request, Reply and Payload. Dashed borders in Figure 20 indicate that the information is optional. The Header field has two required elements called a Noun and a Verb. The Noun identifies the type of the payload and the Verb the type of action. Examples of the possible Nouns are *MeterReading* and *Outage Report*. Nouns are used to identify the information type, which correspond to the XML document defining the structure of the message information [39]. The payload can be described in any other format too in addition to a XML document. However, from the XML document, the XML Schema can be generated easily for the interface implementation.

The standard identifies lists of verbs to be used with each message type. For example with a *request* message verbs GET, CREATE, UPDATE, CANCEL, CLOSE and DELETE can be used and a verb REPLY is used with a *reply* message. More information about the use of the Verb and the Noun fields can be found in IEC 61968-100 standard [33].

According to the CIM standards, payload must be provided for request messages when the verb is CREATE or CHANGE. Also some response and some event messages expect a payload. Payload can contain information in any type, but typically XML documents that conform to a defined XML Schema are used [39]. The payload structure has been designed to support a variety of payload formats and also to compress large payloads. Within the scope of this thesis, payloads stay small enough that it is not necessary to compress them.

4.2 MultiSpeak

MultiSpeak by the National Rural Electric Cooperative Association (NRECA) is the most applied de facto standard in North American distribution utilities. The development of MultiSpeak began already in 2000 in order to establish interoperability among software application interfaces in small electric cooperatives. The use of MultiSpeak is not limited only to small utilities due to its scalability and incorporation of web services [34].

MultiSpeak standard is presented in UML model and use cases are used for defining functional requirements. Use cases are documented as sequence diagrams using Enterprise Architect (EA) software. MultiSpeak is a normative standard and the development trend has been from practical ground level up to general standard. This has led to direct development of profiles, and at the moment MultiSpeak is able to offer around 30 profiles for different operations in electrical grid. Examples of those profiles are meter reading, meter data management, outage detection, load management, demand response and distribution automation control.

MultiSpeak uses web services for real time integration. Web services make use of widely-adopted technologies and protocols such as XML Schema, SOAP, WSDL, HTTTP and XML. These technologies are explained in more detail in Chapter 3.4.

In comparison to the CIM, MultiSpeak is more mature, but covers narrower scope. MultiSpeak provides similar capabilities as IEC 61968 distribution extension in the CIM. In future, MultiSpeak will be developed towards IEC CIM to achieve better interoperability and to reduce the need for developing and maintaining dual interfaces. [35]. The CIM standard also provides wider scope of specification than the MultiSpeak.

5 REQUIREMENTS SPECIFICATION

In software development five steps can be identified regardless of the development methodology. These steps are requirements definition, design, implementation, testing and maintaining. The first step includes capturing requirements for the software. In the scope of this thesis, this means capturing requirements for the interface between the DMS and the WMS. This chapter describes how requirements were collected to construct the interface for the new DMS-WMS integration.

Requirements capture is very critical part of the development process, because the success of the system integration depends on the fact how well the desired functionalities have been captured. If the end product does not complete required functionality, it does not matter how well the following steps are completed. Requirements capturing is an iterative process, because requirements tend to evolve throughout software development. Requirements evolvement must be, however, distinguished from expanding of the application domain. Drawing clear boundaries around the application domain helps to avoid the expanding of the product and helps to concentrate on specifying requirements within the domain. [36]

Requirements capture highlights the real-world goals and it should be able to answer to the question what the end system does. At this point of the process technical details should be ignored. [37]

Requirements can be divided into functional requirements and non-functional requirements. Functional requirements describe services the systems provide to the end users and non-functional requirements describe constraint and standards the end product must deliver. Non-functional requirements include goals for security, availability, safety and performance. Also requirements for testing are listed here. The end result of requirements capturing step is requirements specification that describes the functionalities and constraints of the end product. [36]

5.1 Functional Requirements

This chapter describes the sources and the origins of the requirements for the interface between the DMS and the WMS. The DMS-WMS integration must be able to perform the same functionalities that the existing DMS-WMS integration contains. In addition, new requirements coming from customers and market regulators must be considered. Table 3 describes the requirements and the origins of the requirements for the interface.

Table 3 Functional Requirements for the DMS-WMS interface

		Customer						Reporting
		Add Pictures	Expiration Date	Contact Information	Repair start time	Sending one at a time	Faster functionality	Outage Details
Existing Interface	Condition Report	x	x					
	LVFault			x	x			x
	LVMaintenance			x				x
	MVFault			x	x	x		x
	MVMaintenance			x		x		x
	Outage History						x	

The first subsection introduces the functionalities of the existing interface. The second subsection lists new requirements requested by the customers and the third subsection lists new requirements used in reporting. A more detailed description of the structure of the constructed interface is given in Chapter 6.

5.1.1 Existing Interface

The former version of the ABB MicroSCADA Pro DMS600 and the WMS integration is based on Microsoft Component Object Model (COM) technology. The COM is a technology that enables communication between software components. However, this interface has reached the end of its lifecycle and needs to be renewed. The new interface is required to inherit the functionalities of the existing interface. Six different use cases were identified in the existing interface. These include the Condition Report, LVFault, LVMaintenance, MVFault, MVMaintenance and Outage History.

Condition Reports are used to report the condition of the physical network. Condition reports can be created either in the DMS or in the WMS and existing condition reports can be updated by users of both systems.

LVFault means transferring information about faults occurring in low voltage network. Low voltage faults are created in the WMS based on the customer contacts and sent to the DMS. Existing LVfaults can be updated by users of both systems.

LVMaintenance means transferring information about maintenance work done in low voltage network. Low voltage maintenances are created in the WMS and information is sent to the DMS.

MVFault stands for faults occurring in medium voltage network. The DMS detects MV faults in the network and sends all active MV faults to the WMS according to a timing interval.

MVMaintenance means transferring information about maintenance work done in medium voltage network. The DMS sends all active and planned MV maintenance works to the WMS automatically according to a timing interval.

Outage History means a functionality that a WMS user can send a request to the DMS to get all outages that have occurred at a certain customer site.

5.1.2 Customer Requirements

Customer requirements mean requirements that originated either in DSOs or in system vendors. Also the previously conducted data mapping for work order management in SGEM program was considered in customer requirements. Data mapping to transfer information between a DMS and a WMS had been started in work package 6 in SGEM program already in 2008. This data mapping is a draft in SGEM portal [13] and therefore, it is not directly applied in this interface. However, it is considered as a source and reference point to avoid duplicate work. In addition to those parameters found in the existing interface, this document lists parameters for transferring information about fault location, type and reason like stated in the new regulatory requirements above. In addition to parameters used for describing fault, this document lists parameters for sharing contact information of the work group and control center personnel. Transferring contact information in the interface was requested by DSOs, because it helps the control center and work crews to communicate with each other during a fault situation.

Adding a picture to a condition report was requested by customers. Work group can take a picture at the site and attach that to the condition report. When the condition report is sent to the DMS also the control center personnel can view this additional information. This functionality provides information to the DSO to prioritize the destinations that require maintenance.

Another new requirement considering Condition Reports was a possibility to add an expiration date to the report. Expiration date indicates that the condition report should be handled by that date or that information is invalid after expiration date.

Adding a repair start time for a fault is used when the fault reparation work is delayed for a reason independent from the network operator. For example, when weather conditions prevent work group accessing the fault site. These situations are also called as force majeure.

A new functionality of sending one MVFault or one MVMaintenance at a time was requested by customers. In the existing version of the system integration MV faults

and MV maintenance works have been transferred from the DMS to the WMS all at once according to a defined time interval. This has turned out to be fairly heavy and impractical solution and thus the new interface is required to be able to transfer MV faults and MV maintenance works also one at a time when necessary. When a new MV fault or MV maintenance work is created or updated, its information is transferred to the WMS. Also the functionality to send all active MV faults and MV maintenance works with timing is still required, but this new feature enables to elongate the timing interval and thus reduces the amount of data transferred between the two systems.

A faster functionality is requested for Outage History. The existing functionality is asynchronous, which means that when the WMS sends a request to get outages that have occurred at a specific customer site the request is not replied immediately. In order to make this functionality faster the asynchronous requests are changed to synchronous requests that are handled and replied to immediately when received by the DMS.

5.1.3 Reporting Requirements

Besides all other functionalities, the DMS is also used as a reporting tool in DSOs. The integration with the WMS is capable to provide information to the DMS that helps to improve both internal and external reporting. For that reason, also reporting requirements are considered when designing the interface for the DMS-WMS integration.

In Finland, network operators are obligated to report interruptions in electricity distribution to the Finnish Energy Industries (ET) and to the Finnish Energy Market Authority (EMV). Outage reporting requirements for the ET and the EMV differ and the last time the ET requirements were updated was over 10 years ago. Since then the business has developed remarkably and information systems are able to provide more specific outage reports than ever before.

During this thesis work, updating the outage reporting requirements for ET was in still progress. According to the update plans the main changes to the requirements will be that outage reporting becomes mandatory for all voltage levels, including low voltage network and that reports are based on usage points instead of substations like before. According to these new requirements, DSOs need to be able to provide outage reason and outage time to their customers regardless of the reason or the date and time the outage occurs. Additionally, new reporting requirements contain a list of parameters to describe the outage type, reason, and location. This list is provided in Appendix A of this thesis. As this list is still a draft and the DMS600 is also used in other countries beside Finland, where regulatory requirements differ from the ones listed here it was considered best not to hard code these parameters to the interface. These parameters are taken into account when designing the interface so that it is possible to transfer this kind of information between the DMS and the WMS, but it is not considered as mandatory information. According to ET's announcements, the updated version of outage reporting is scheduled to be in use in Finnish DSOs in the beginning of year 2015. Hence, the updated outage reporting data can be used in the next EMV supervision period (2016-

2019). The aim of this update is also to unify the outage reporting requirements for the ET and the EMV. [39]

5.2 Non-functional Requirements

Non-functional requirements for system integration include goals for security, availability, safety and performance. In this thesis non-functional requirements are defined in terms of information security. Also testing requirements are discussed here.

Information security aims to protect information systems and the data they contain from unauthorized parties. Information security consists of multiple factors that are system specific and evolve with time. Thus, security of an information system has to be planned with the system itself and it is an ongoing process that has to be updated continuously. Ideally, information systems would be developed from the security point of view so that security is built into the system rather than trying to add security to it afterwards. One essential method to ensure information security is to use up-to-date software. Following this easy guideline in electricity network operation can be difficult, because systems are operated continuously and the equipment life span in electrical grid is closer to decades than years. Updating systems is challenging and has to be planned carefully. Also the system environment and the nature of information handled affect the level of needed security. Security issues of an isolated system are significantly different from a system exposed to outside a company, integration techniques affect the information security design and information classified as sensitive must be secured better than non-sensitive information. Here, information security study has been limited to the level of the DMS-WMS integration. More specific study of information security in electrical grids can be found in Eerola's master thesis in [12].

The information security can be defined in terms of confidentiality, integrity and availability of information. This information security triad is often referred as CIA.
Confidentiality defines access to the information only to authorized parties. Information cryptography and access controls are used to ensure the confidentiality [40]. For the integration of the DMS and the WMS the confidentiality level of the transferred data is classified as low. The DMS and the WMS systems both contain confidential information, but no confidential information is transferred between the systems. Instead, most of the information is actually transferred to be published for example on a DSO's website in order to provide information for customers.

In this system integration data integrity is more important than the data confidentiality. Integrity defines the trustworthiness of the information. It does not only cover the integrity of the information itself, but also the integrity of the origin of the information [40]. This system integration has to ensure that the data does not change on the way and that the data is received from a trusted source. Violation of data integrity can in worst case scenario result in longer outage times and therefore causes higher outage

costs. This could happen for example if a work crew is instructed to go to a wrong location.

Availability defines the level of accessibility of the system or data. Critical components require higher availability, because they need to be always accessible by authorized users [40]. High level of accessibility is also required in situations where resending information or controlling commands is prohibited. In this system integration resending information is possible, but it is preferred that the DMS and the WMS are accessible at all times. Information about faults and active maintenance works requires higher level of availability than condition reports or upcoming maintenance works. If the connection between the two systems is lost for some reason, phone calls can be used as a backup to transfer the most critical information between different parties. This system integration does not require the highest level of availability, but defining the availability level too low is not desirable either. In a case of fault this system integration provides necessary information for fault repair.

Testing requirements for the system integration of the DMS and the WMS included module testing, integration testing and user testing. Module testing for this system integration means that both systems, the DMS and the WMS, are tested separately with the new interface. This is necessary, because the DMS and the WMS are produced by different vendors. Module testing ensures that the systems are able to send and receive information with this interface. In integration testing the modules from both vendors are tested together. This phase of testing ensures that the two systems are able to communicate with each other over the interface. When the module testing and integration testing are completed successfully user testing can begin. User testing ensures that the system integration functions as required. This means that changes the user does in the user interface are transferred to the other system and showed in the other system's user interface correctly. If possible, it is preferred that users who regularly operate the DMS and the WMS participate in user testing. Then the testing is as close to real operation as possible. When the user testing is completed successfully the system integration can be delivered to the end customer.

6 USE CASES

In software engineering use cases are used to present functional requirements of a software system. Use cases are determined in a very abstract level and technical details are ignored at this point. In addition to textual description different kinds of diagrams are often used to visualize use cases. Diagrams provide a common ground for different stakeholders and help people with different technical backgrounds to understand each other. Well defined use cases provide a good starting point for the implementation.

This chapter introduces use cases that have been defined based on the previously listed requirements for the integration of the DMS and the WMS. Each use case is first described generally and then data flow between the systems is presented in a sequence diagram.

6.1 Condition Report

Condition report is used to record findings in the network that can lead to interruptions in electricity distribution. Condition report is an important tool in network maintenance as risky conditions in the network can be repaired before they actually cause interruptions. Use of condition reports also ensures that the information about network condition is not lost as it is recorded to the system. Condition reports can also be used to approximate fault location if it is not otherwise known. For example, in a LV fault that does not have an exact location the DMS user can map all condition reports to see if there has been any reports in the fault area.

Condition reports can be created and updated either by WMS or DMS users. Sequence diagram in Figure 21 presents a possible sequence of events for condition report life cycle. In this example condition report is created by a WMS user.

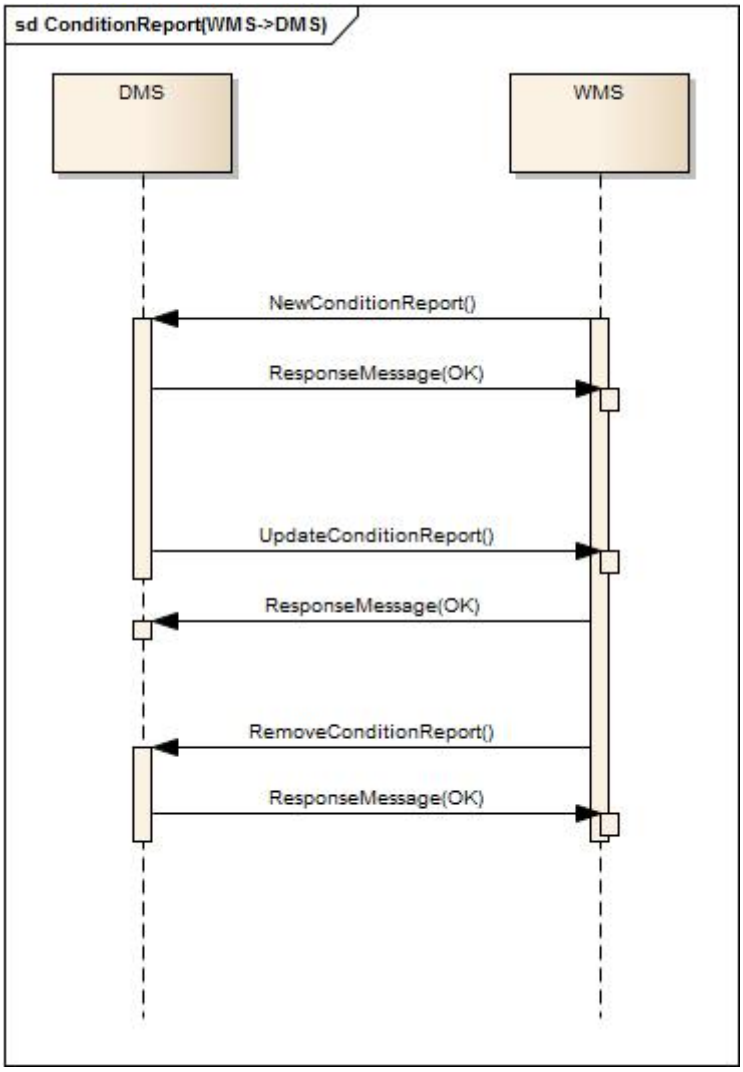


Figure 21 Condition Report

New condition report is created in the WMS for example by a work group. A request to add new condition report is then sent to the DMS, which sends a response message with OK status back to the WMS when adding the report has been successful. This new condition report is then presented in the user interface of the DMS along with all other condition reports that are saved to the database. The DMS user can modify and add information to the condition report and these changes are sent to the WMS. The WMS also sends a response message with OK status when updates have been successful. When the condition in the field has been cleared or it is not valid anymore, the report is removed by the WMS user and *RemoveConditionReport* message is sent to the DMS. The DMS removes that report from the database and replies back to the WMS.

Figure 22 presents the CIM profile used to transfer information required for condition report.

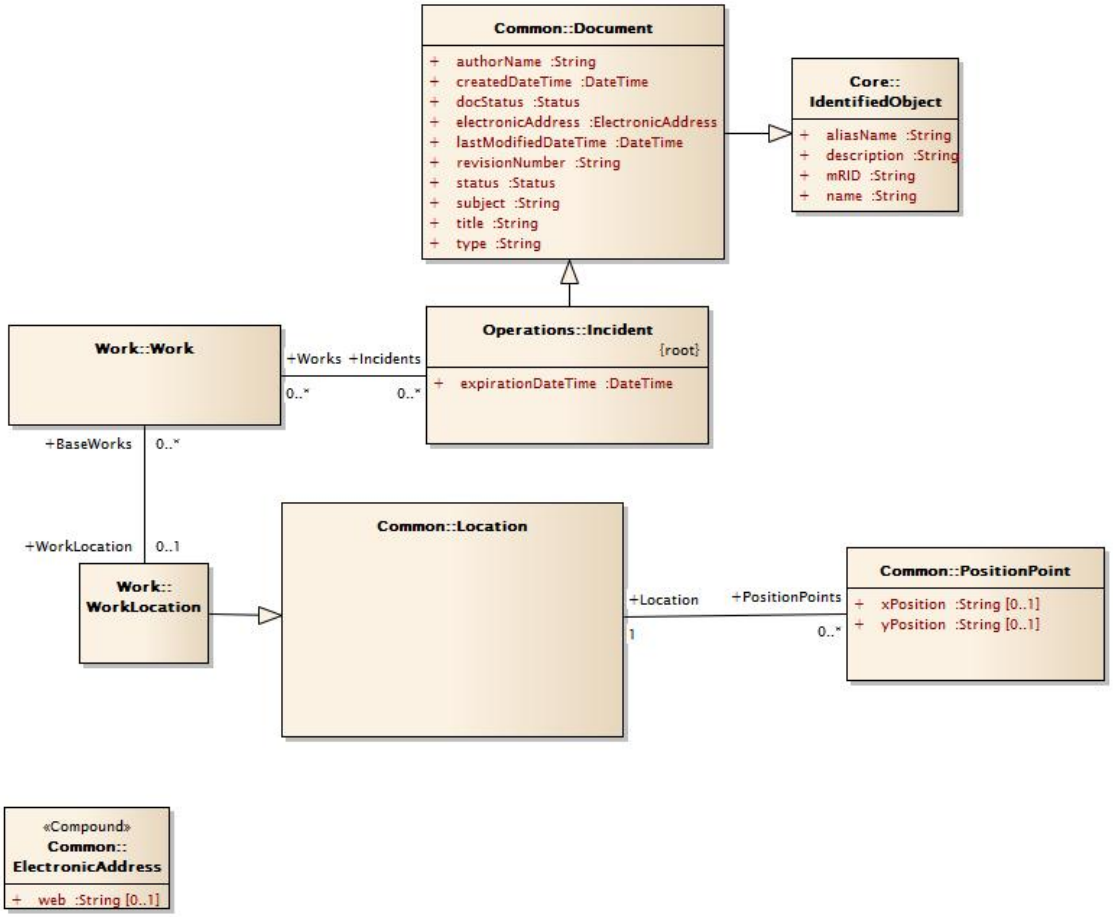


Figure 22 CIM base profile for Condition Report

Table 4 below lists the used parameters in a table format to clarify the inherited parameters.

Table 4 Parameters used in Condition Report

CIM Parameter	Type	Definition
Incident::mRID	string	Report ID
PositionPoint::xPosition	string	Location x
PositionPoint::yPosition	string	Location y
Incident::createdDateTime	dateTime	Create Date of Report
Incident::description	string	Report Description
Incident::lastModifiedDateTime	dateTime	Resolved Date of Report
Incident::subject	string	Company name
Incident::expirationDateTime	dateTime	Expiration date for the report

Incident::mRID and Incident::*createdDateTime* are mandatory information for all condition reports and must be filled when creating a new condition report. *Inci-*

dent::expirationDateTime on the other hand is not compulsory information and can be filled when needed.

6.2 Low Voltage Fault

Low voltage (LV) faults are unplanned interruptions in the electricity distribution occurring in the low voltage network. Low voltage network is defined to cover voltage levels from 0.4 kV up to 1 kV. Faults occurring in the network can be divided into long interruptions, which means interruptions over 3 minutes and short interruptions that last less than 3 minutes. Short interruptions are mainly reclosing operations, which do not require further actions and are therefore defined to be out of the scope of this thesis. In this context, word 'fault' is used only to describe long unplanned interruptions.

Due to the prevalence of AMR meters, LV faults can nowadays be detected automatically. Before AMR meters, customer contacts were the main way to detect LV faults. However, AMR meters are fairly new technology in many DSOs and best practices to use the information they provide are still forming. Some operators react to all AMR alarms whereas others have lists of alarm codes they react to and some wait for customer contact before further action.

A LV fault can be created either by the DMS or the WMS. In a case the DMS creates a new LV fault the impulse most likely comes from an AMR meter. According to the regulation accepted by Finnish Government, eighty percent of every distribution operator's customer sites must have a remotely readable electricity meter by the end of year 2013 [16]. Despite the fact that this goal has been reached ahead of time, not all of the customer sites still have an AMR meter installed. For this reason, functionality to create LV faults manually is required. LV faults must be created manually also in situations where the circuit-breaker does not open automatically. Manually created interruptions are classified as faults, when customers are not informed in advance enough so that they would have time to prepare for the interruption in electricity supply.

Figure 23 shows a sequence for the process of LV fault creation by the WMS. A new LV Fault is created based on, for example a customer contact. Fault details are sent to the DMS. If new information about the fault is obtained, both the WMS and the DMS users can update the fault information. When the normal situation has been attained, the WMS reports back to the DMS by sending a fault over message. Fault over message can also be sent from the DMS to the WMS if the field crew informs the control center by phone about the fault repair. When the fault has been repaired, control center uses the DMS to calculate fault statistics for reporting.

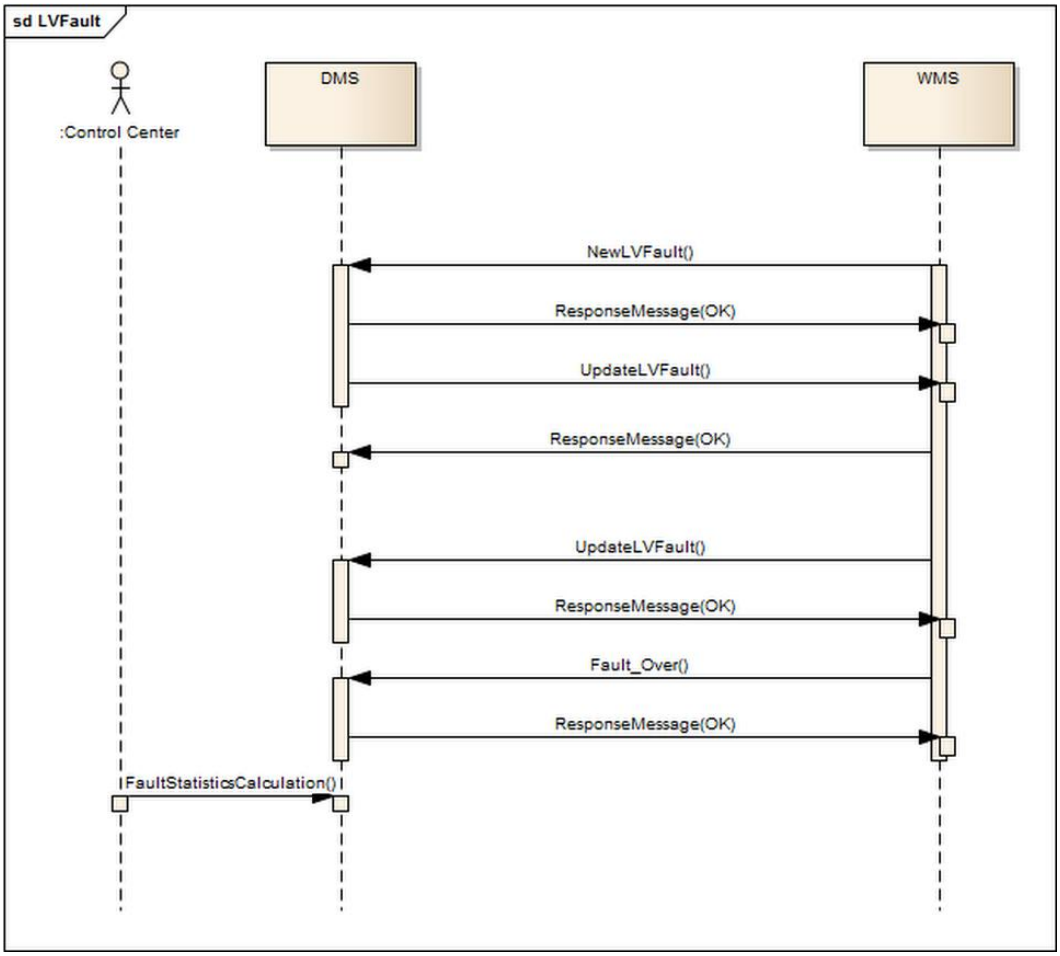


Figure 23 Low Voltage Fault

The class diagram for LV Fault is more complex than the class diagram for the condition report as can be seen in Figure 24 below. More classes and class relations are required to present the information transferred in a case of a LV Fault.

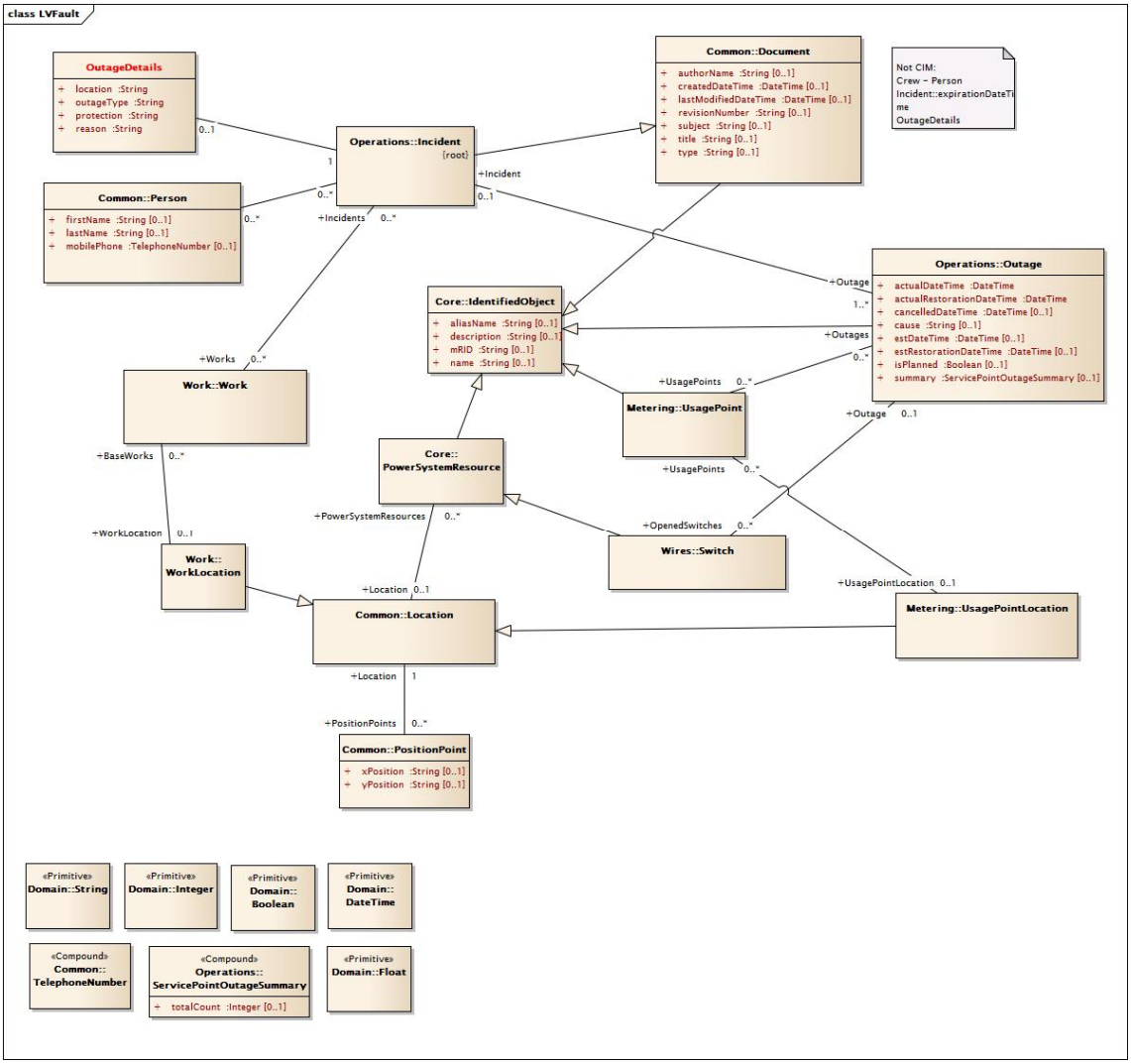


Figure 24 Class diagram for LV Fault

Necessary parameters for transferring LV Fault information between the DMS and the WMS are listed in Table 5.

Table 5 Parameters used in LVFault messages

CIM Parameter	Type	Definition
Incident::mRID	string	Report ID
Incident::description	string	Report Description
Incident::name	string	Name for the interruption
Incident::authorName	string	Person responsible for the interruption in the control center
Outage::type	string	Type of the interruption. Here LVFault
Outage::actualDateTime	dateTime	Start time of the fault
Outage::actualRestorationDateTime	dateTime	End time of the interruption
Outage::estRestorationDateTime	dateTime	Estimated end time of the interruption
UsagePoints::mRID	list	Customers affected
OpenedSwitches::mRID	string	LV Feeder ID
OutageDetails::Location	string	Location of the maintenance, substation, overhead lines, cables etc.
OutageDetails::Type	string	Describes if the fault occurs in own network or in customer network
OutageDetails:Reason	string	Reason that caused the interruption
PositionPoint::xPosition	string	Location x
PositionPoint::yPosition	string	Location y
Person::name	string	Name of the person in the field
Person::mobilePhone	string	Contact number for the person

Incident::mRID that contains a unique identifier for the fault and *Outage::actualDateTime* stating the starting time of the fault are mandatory information when creating a new fault. Other parameters can be filled in later when relevant.

6.3 Low Voltage Maintenance

Low Voltage Maintenance is an interruption in electricity supply in low voltage network that has been planned in advance enough to inform customers. For LV maintenance starting time and switching plan are known in advance unlike for LV fault. LV maintenances are also separated from LV faults for reporting purposes.

LV maintenance request can be created either by the DMS user or the WMS user. Figure 25 describes a sequence of LV maintenance created by the WMS user.

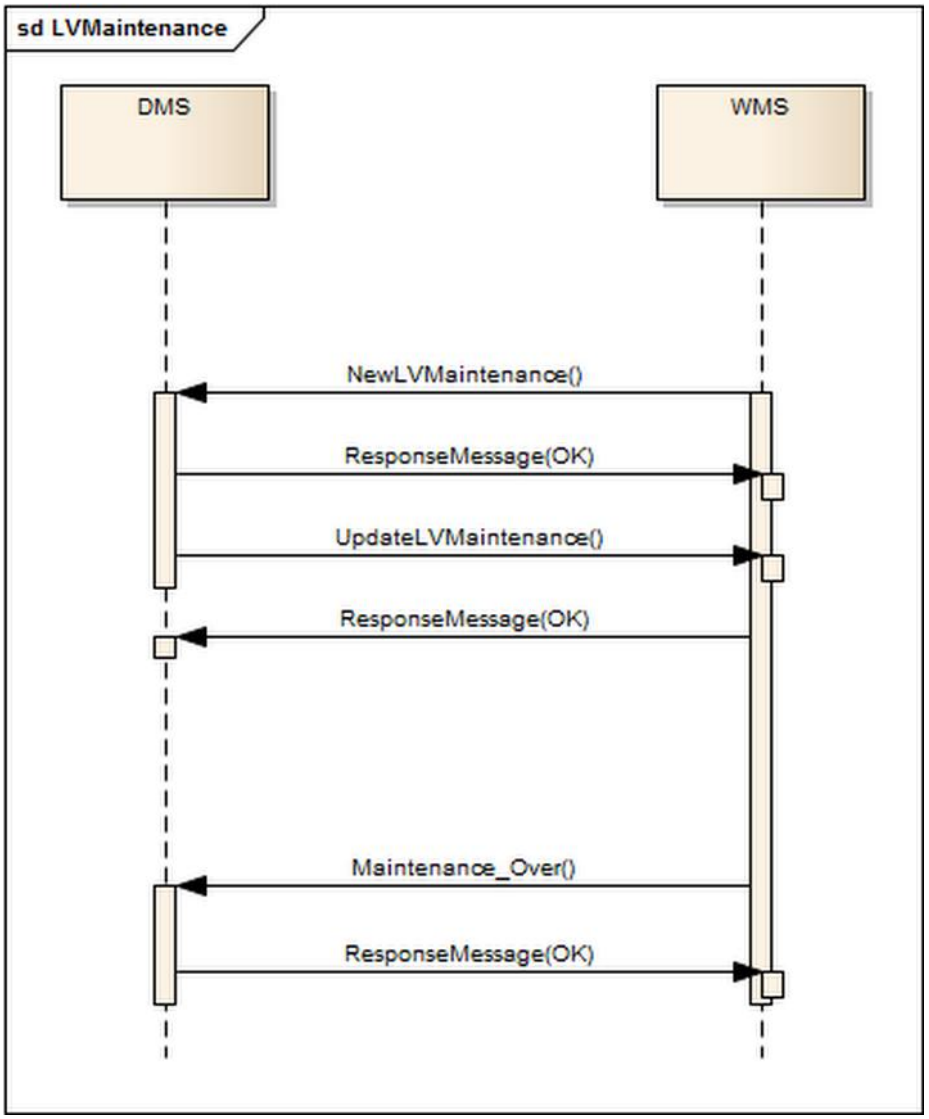


Figure 25 Low Voltage Maintenance

A WMS user creates a new LV maintenance which is then transferred to the DMS. The control center handles the new maintenance request and updates the maintenance for example by confirming a start time for the maintenance work. The DMS and the WMS users are both allowed update the LV maintenance. Based on these updates customers can be informed if the repair time is extended and the control center knows the situation in the field. When the maintenance work has come to an end the work crew reports to

the control center either by sending maintenance over message from the WMS or by calling to the control center. Practices vary from DSO to another.

Only the customers that experience an interruption in their electricity supply because of the maintenance work are informed. LV Maintenance can sometimes be carried through so that electricity supply is maintained to all customers during the work. This can be done for example by using reserve power.

Parameters used for transferring LV Maintenance messages are the same as used for LV Fault with an exception that *Outage::type* is LVMaintenance instead of LVFault.

6.4 Medium Voltage Fault

Medium Voltage (MV) Fault covers faults occurring in voltage levels from 1 kV up to 100 kV. The DMS-SCADA integration is in response for detecting MV faults in the network. When a circuit breaker opens at the substation, the SCADA reports the change in switching state to the DMS. First, an auto reclosing operation is launched to clear the fault. If the circuit breaker stays open regardless of the fast enclosing operation a new fault is created in the DMS. Creating a new MV fault manually in the DMS is also possible. This is, however, used only when connection with the SCADA is down or when demonstrating a fault in a simulation purposes.

The DMS updates the network image according to the new topology, calculates the most likely fault locations and adds the fault to a list of faults. Next the fault is isolated and electricity supply restored to as many customers as possible in order to minimize outage costs. The DMS offers support for the fault isolation. When the fault zone is isolated by operating the remotely controlled and manually operated disconnectors, the work crew may start the reparation work.

As the reparation work proceeds, the fault zone can be reduced and electricity supply recovered to a wider area. When the electricity supply has been returned to all customers fault is marked as repaired.

Figure 26 demonstrates the communication between the DMS and the WMS in a MV fault.

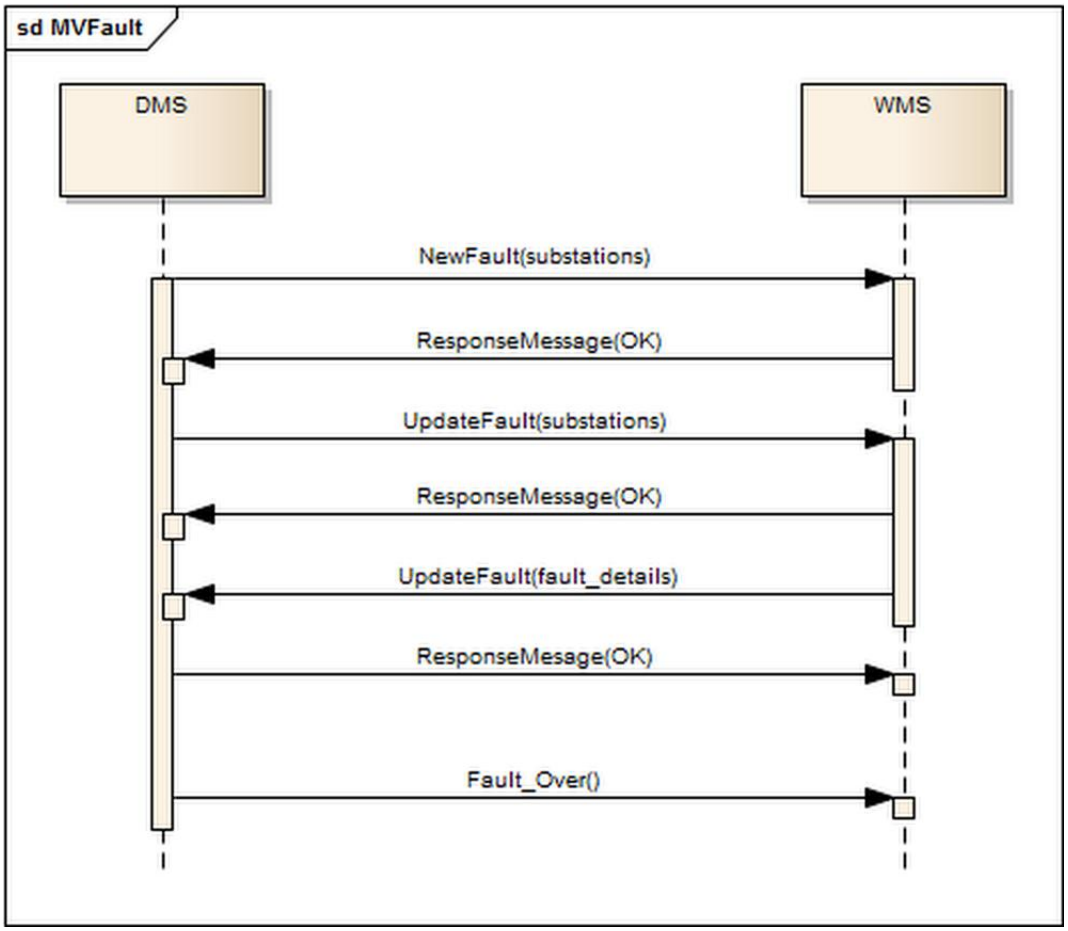


Figure 26 Medium Voltage Fault

Those faults that cannot be cleared by auto reclosing operation and require repairing are sent to the WMS. First a new MV fault is created in the DMS and sent to the WMS with information of all substations affected by the fault. When the fault is isolated, the DMS sends fault location information and updated substation list. The list contains those substations without electricity at the moment and those that will be affected at some point of the reparation process. The update message is sent every time there is new information to be updated to the WMS. The WMS user can update the fault details such as fault reason. Examples of possible fault reasons are snow, thunder, animals, and structural defect. This type of information is often available only at the fault site. A detailed description of used outage parameters is listed in Appendix A. When electricity is restored to all customers, the DMS sends a fault over message to the WMS.

In a MV fault it is crucial that the network switching state is constantly known by the control center personnel. MV fault is assigned to one person in the control center who is then responsible for taking care of that fault. This procedure ensures that at least one person is always aware of the situation and the work crew can directly be in contact with that person in the control center.

Profile for transferring MV faults is very similar to the profile presented in LV fault section. However, some differences exist and MV and LV faults should be separated from each other because of the different nature of the faults. The procedures to repair faults are distinct in different voltage levels. Figure 27 below presents the profile used for transferring MV faults.

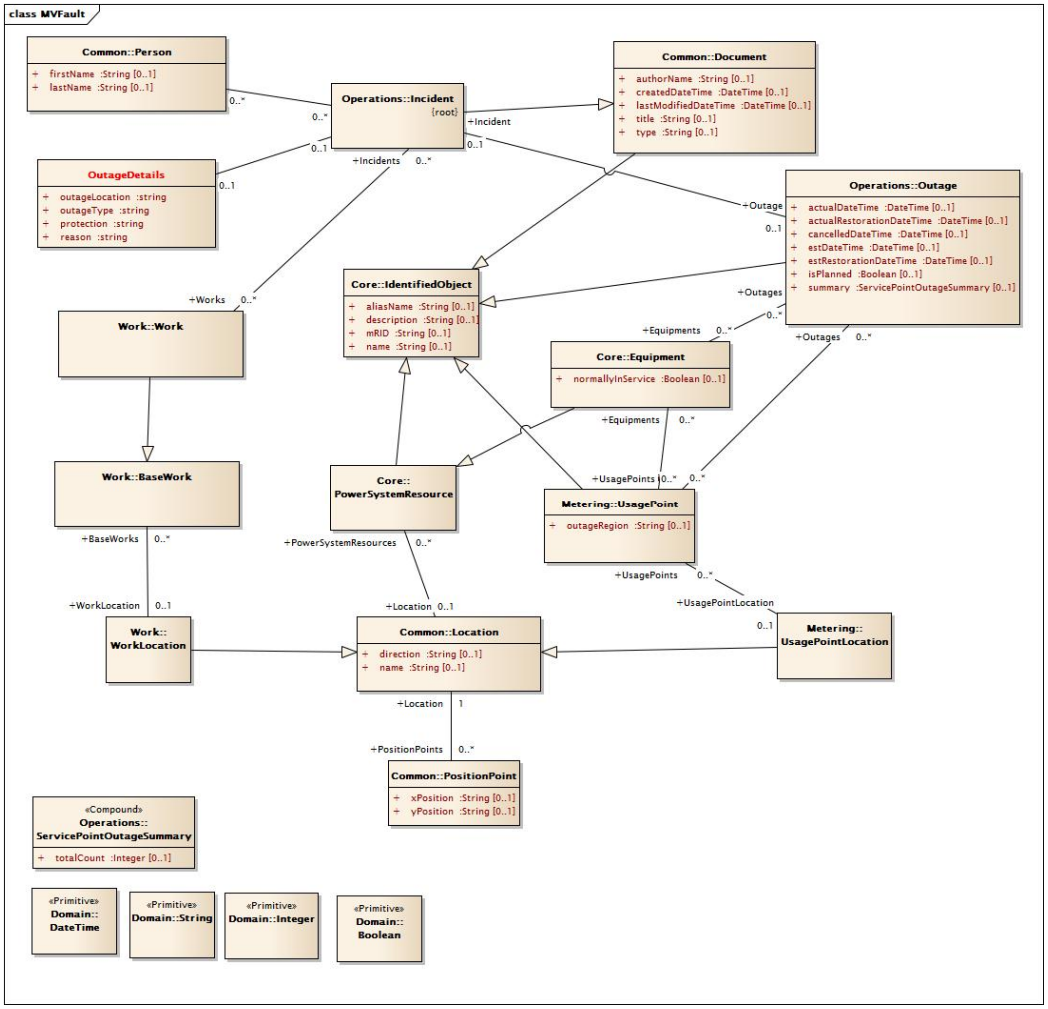


Figure 27 Profile for Medium Voltage Fault

The main difference with the profile used to transfer MV faults and MV maintenances in comparison to the profile used for LV faults and LV maintenances is the *Core::Equipment* class, which is used to describe substations included in the MV fault or MV maintenance. A parameter called *normallyInService* in the *Equipment* class describes if the substation is without electricity at the moment. The *Switch* class that is used to present the LV feeder isolating the fault area in LV fault is not used for MV faults or MV maintenances.

6.5 Medium Voltage Maintenance

Medium voltage maintenance is a situation when maintenance work in medium voltage network causes an interruption in electricity distribution. Maintenance work that does not cause an interruption in electricity supply is defined to be outside of this use case and is defined to be outside of the scope of this thesis. Maintenance work is planned in advance and the switching plan is tested to make sure that voltage levels stay acceptable and protection is sufficient. These checkups can be done with the DMS simulation.

When the switching plan is completed, authorized DMS user accepts the plan and the plan is added to the switching plan management list. Unlike with the MV fault, the start time of the maintenance work is known in advanced and the end time can be estimated. This enables DSOs to inform their customers in advance.

Figure 28 describes the use case for MV maintenance. In addition to the information flow between the DMS and the WMS also the control center personnel is presented to clarify what event gives an impulse for which message sent to the WMS.

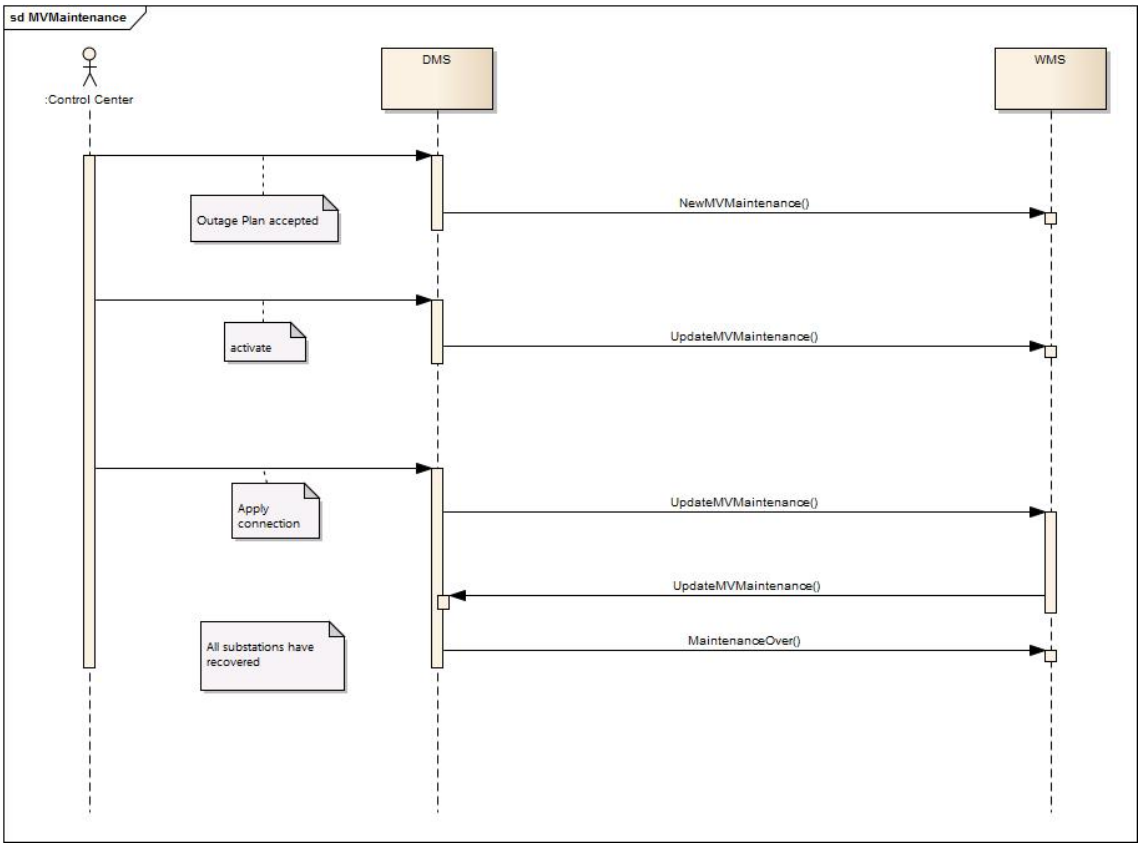


Figure 28 Medium Voltage Maintenance

A new maintenance plan in sent to the WMS when the switching plan is accepted by the authorized DMS user. When the plan is activated in the DMS and every time the switching state is changed an update message is sent to the WMS. Maintenance details can be updated also from the field by sending update messages from the WMS to the DMS. These update messages may, for example contain updates for estimated ending time or

extend of the maintenance work. When normal switching state is restored and the field crew informs the control center that their work is completed maintenance over message is sent to the WMS.

In MV maintenance work it is crucial that control center knows the actual switching state of the network at all times. For this reason, field crew and control center personnel responsible for the maintenance work must be in contact with each other through the work.

The profile used for transferring MV maintenance messages is the same as used for MV faults with an exception that *Outage::type* is MVMaintenance instead of MVFault.

6.6 Outage History Query

Outage history query is used to obtain all outages that have occurred at a customer site. Figure 29 describes the message exchange between the WMS and the DMS in this use case.

Figure 29 Outage History Query

The WMS sends an outage history query to the DMS with the connection point id as an input parameter. The DMS returns a list of outages that have occurred at the site. This use case enables the system operator to view the outages occurred at specific customer site. The query can also be specified to consider only outages caused by faults or maintenance works or outages in low voltage or medium voltage network.

7 PROFILE

This chapter describes the profile used in data exchange between the DMS and the WMS. The profile is constructed based on use cases and class diagrams described in Chapter 6. Figure 30 below presents the profile in an UML class diagram. The same diagram is enlarged in Appendix B.

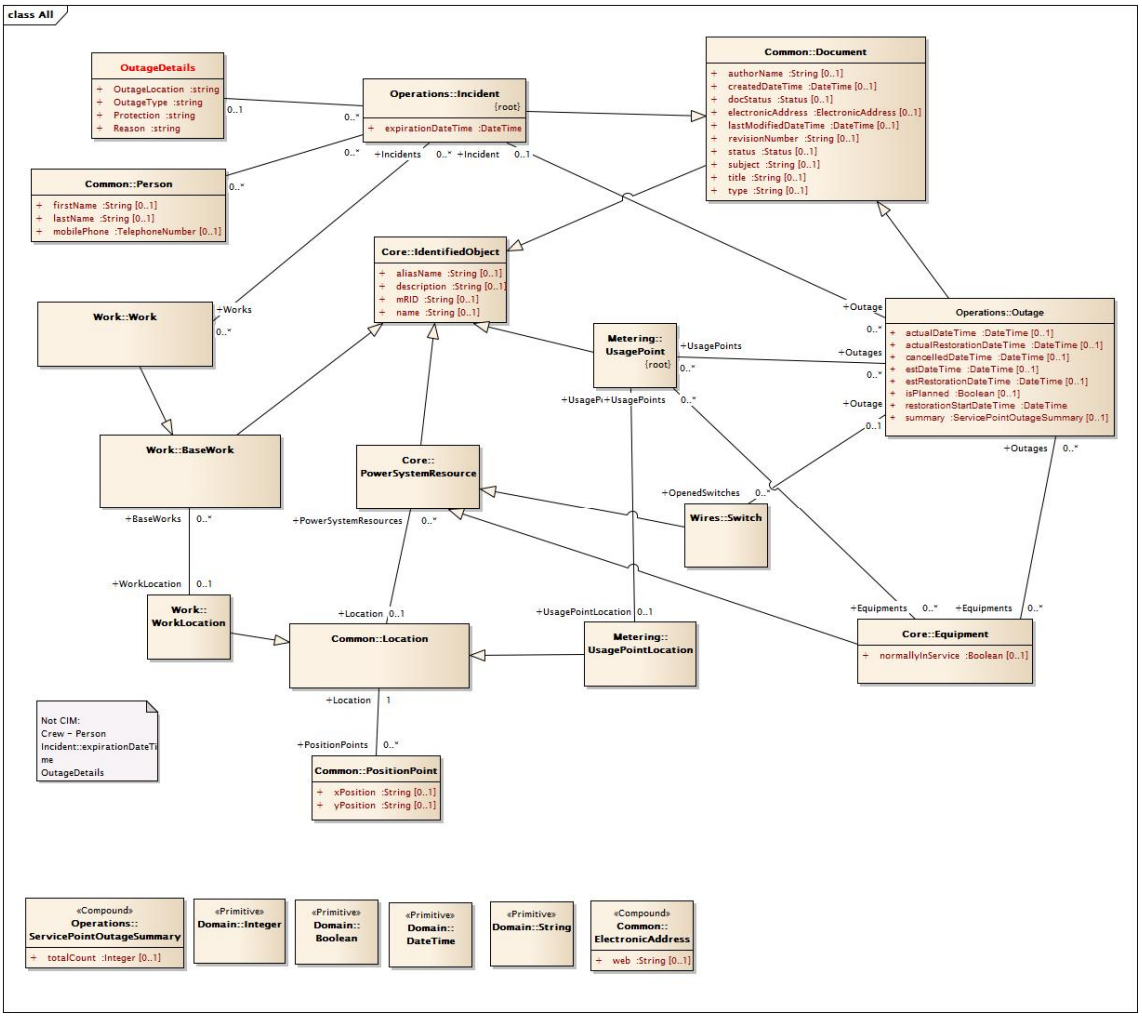


Figure 30 Profile used in the Interface

The CIM standard is used as a basis in constructing the profile for integrating the DMS and the WMS. The aim was to keep the changes to the CIM model minimal in order to be able to evaluate the suitability of the standard for this system integration. However, some alterations and additions to the standard were necessary in order to create a profile that satisfies the functional requirements set to the interface. These changes are described in the following subsections.

7.1 Class Relations

A bi-directional association exists between the *Incident*-class and the *Outage*-class in the CIM standard. According to the standard, multiplicity values for this association are as described in Figure 31.

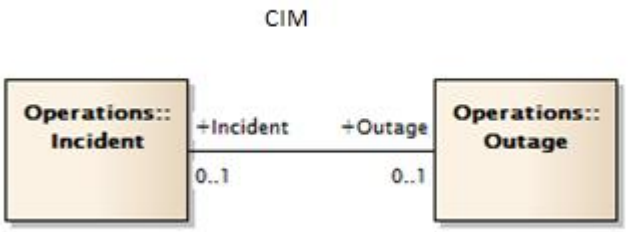


Figure 31 Multiplicity values of the Incident class and the Outage class according to the CIM standard

The multiplicity value next to the *Outage* class is 0.. 1. It means that an *Incident* instance can have either one *Outage* instance with it or no *Outage* instance at all. This same applies also the other way around; the *Outage* instance can have either one or none *Incident* instance.

The multiplicity value of the *Outage* class causes problems from the interface point of view when an interruption consists of more than one outage steps. Outage steps occur with faults and maintenance works. For example, in a case of fault the first outage step begins when the fault occurs and ends when the fault is isolated to a specific area and the electrical state of the network changes. New outage step is created every time the switching state is updated and electrical state of the network changes until the whole fault has been repaired and the fault is over.

The system integration for the DMS and the WMS can be done with the CIM model without altering this property, but then only one outage step can be transferred with the interface at a time. The overall outage that consists of several outage steps has to be constructed in the DMS and in the WMS internally. In order to transfer the overall outage with all outage steps that belong to it, the CIM standard model was moderated to look like described in Figure 32.

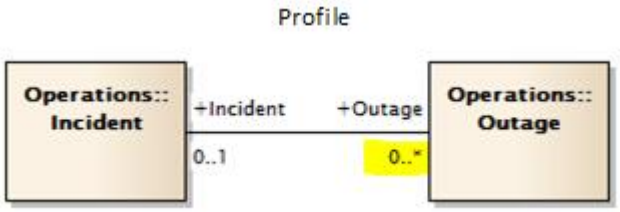


Figure 32 Multiplicity value fixed for the profile

By changing the multiplicity of the *Outage* class to 0..*, enables transferring faults and maintenances that include outage steps with the interface. At the moment this functionality is not absolutely necessary, but if the overall outage including all outage steps is required to be sent from the DMS to other system this functionality is required. Earlier versions of the CIM standard have had a class called *OutageStep* to describe this functionality, but it has been removed from the current version of the CIM standard (iec61970cim16v17_iec61968cim12v06_iec62325cim02v07). Adding a new class called *OutageStep* would have been an alternative approach to enable transferring outage parts in this interface. However, changing the multiplicity of the Outage class was found to be easier and required fewer alterations to the current CIM standard.

7.2 Outage details

An *OutageDetails* class was added to the profile in order to transfer the parameters listed in Appendix A. These described parameters are specific to Finnish energy market regulators. In order to keep the interface as general as possible, these parameters were not hardcoded to the interface but instead defined as type *string* which allows the end systems, in this case the DMS and the WMS, to decide what values are transferred in the interface. Hard coding parameters to the interface would lead to a need to update also the interface when used parameters change.

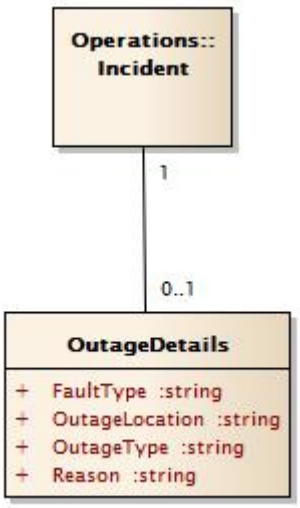


Figure 33 Association of the Incident class with the OutageDetails class

The *OutageDetails* class is associated with the *Incident* class, because outage details are common to all outage steps within the same overall outage. As described in Figure 33, the *Incident* class can have zero or *one OutageDetails* instance and the *OutageDetails* can have one *Incident* instance.

7.3 Incident - Person association

The interface is required to be able to transfer contact information of a person working the field in a case of fault and maintenance work. Figure 34 describes how the contact information of a person is included to the profile when following strictly the CIM standard. The *Person* class contains required parameters such as person's name and phone number to be transferred between the systems.

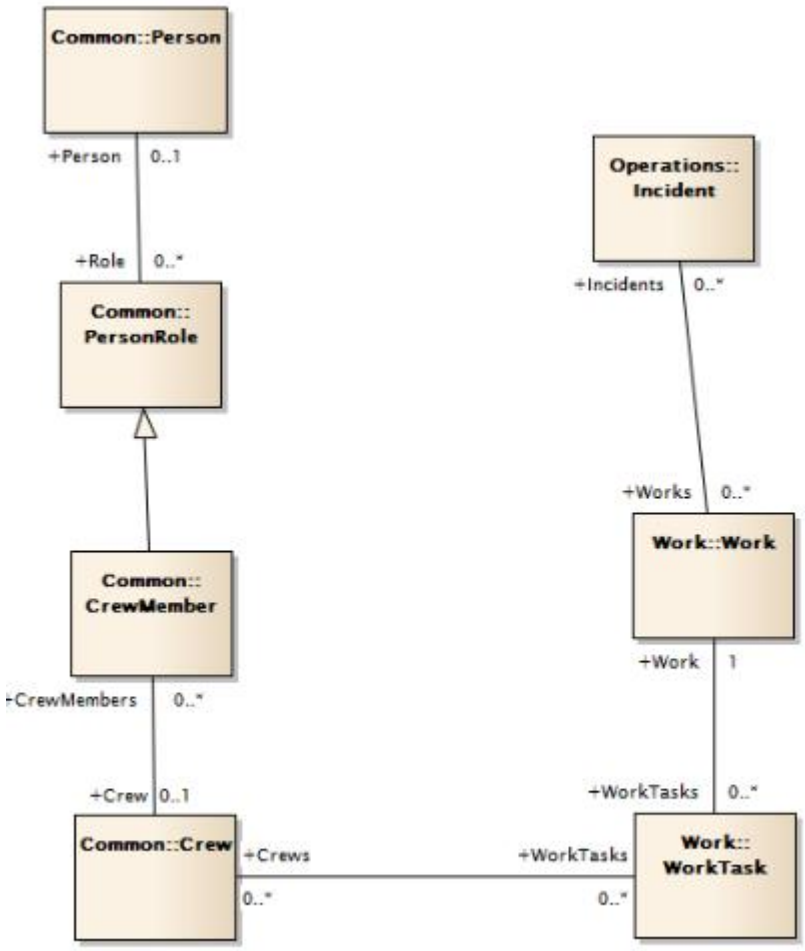


Figure 34 Including contanct information of a person according to the CIM standard

When the *Person* class is added to the profile according to the CIM standard documentation, problems appear with the XML Schema generation. Class relations prevent the *Incident* instance from obtaining the contact information of a person and because the code is generated based on the XML Schema file the same problem is transferred to the code. The *Incident* instance is used to describe the maintenance work and the overall fault and therefore the contact information has to be traceable from the *Incident* instance. With this model, the contact information cannot be obtained for the *Incident* as required.

One option to solve this inconvenience in the profile is to alter the class relations manually in the code. These alterations violate the CIM model and changing the code manually easily leads to errors and complicate the maintenance of the interface. Thus, other options to include contact information of a person to the profile were examined.

First, the profile was changed so that the *Person* class was directly associated with the *Incident* class. This is presented in Figure 35.

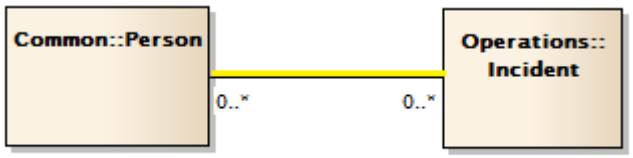


Figure 35 Direct Association between the Incident-class and the Person-class

This enables to include contact information with an *Incident* instance so that contact information is traceable starting from the *Incident*. This solution fulfills the requirements set to the interface and in a normal situation when fault has a specific location where the person or work crew is assigned to go this solution works as needed. In a deviant situation such as a major disturbance, when one fault is divided into several separate tasks assigned to different people this model is too simple to associate the task with the right person. To include this functionality to the interface the profile was altered as presented in Figure 36.

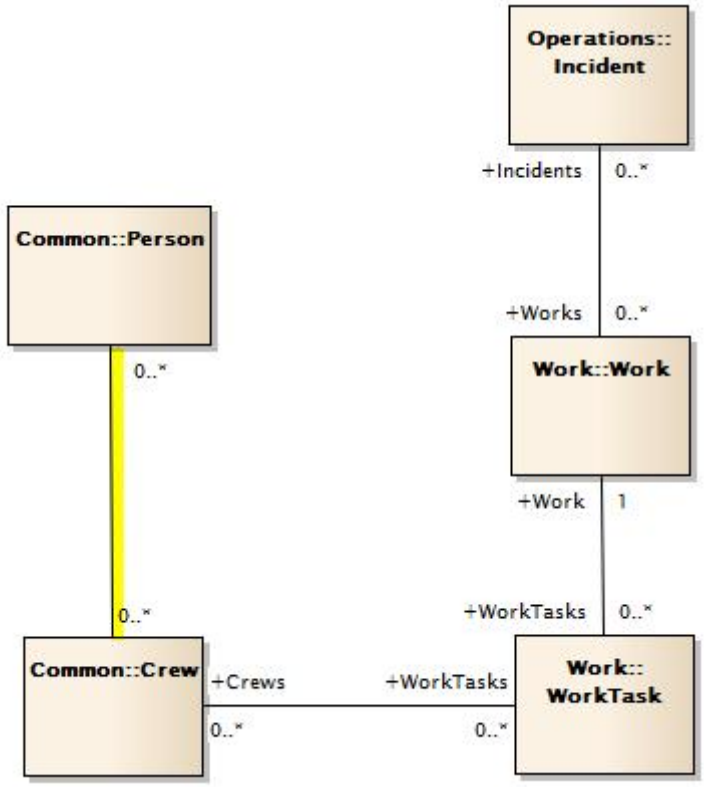


Figure 36 Association added for the Person-class and the Crew-class

The association that is not part of the CIM standard is highlighted with yellow. This model enables transferring contact information of several people assigned to different tasks. Location parameters are associated with the *Work* class and therefore by using this model also the location assigned to a person can be traced. This model provides enough functionality for a deviant situation, but does not complicate the profile for a normal use either. Generating XML Schema from this model does not require any alterations to class relations at the code level and that is why this model was chosen to be used in the profile.

7.4 Presentation of Substation

The interface is required to be able to transfer information of substations related to a MV fault or to MV maintenance. The CIM model provides a *Substation* class to present information related to substations. According to the current CIM standard the *Substation* class relates to other classes used in the profile as presented in Figure 37.

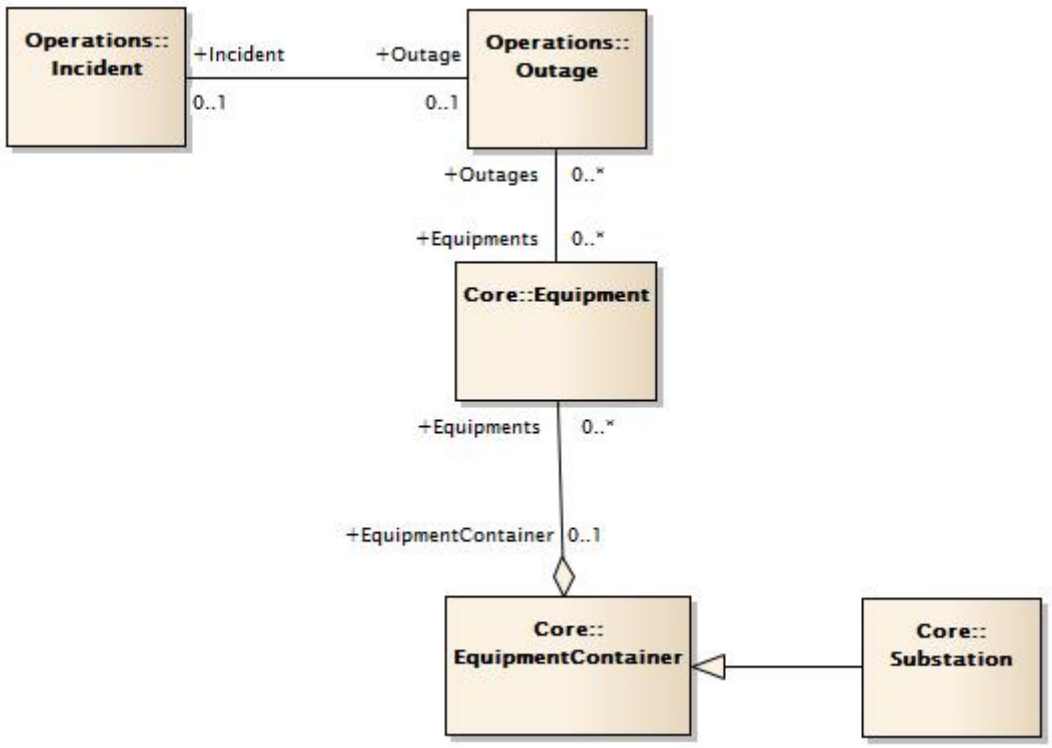


Figure 37 Class relations to include Substation class to the model

The *Outage* class is associated with the *Equipment* class in a way that an *Outage* instance can have multiple *Equipment* instances. The *Equipment* instance again can belong to an *EquipmentContainer* instance. The *Substation* instance inherits from the *EquipmentContainer*. According to the documentation of the CIM standard version used here, the *Equipment* class is used to describe physical devices of a power system. When associated with an *Outage* instance, *Equipment* class describes the state of the equipment with *normallyInService* value.

The problem with this model is that *Equipment* is not aware of the *EquipmentContainer* it belongs to. The aggregation association between the *Equipment* and the *EquipmentContainer* instances, described as an unfilled diamond, works so that only the *EquipmentContainer* knows its *Equipment*s. Therefore it is not possible to know which substations are associated with an outage instance. If the *Equipment – EquipmentContainer* association was changed to bidirectional association, the *Equipment* would become aware of the *EquipmentContainer* it belongs to. This change alone would

not be enough to solve which substations belong to an outage, because the DMS does not easily provide information of equipment belonging to a fault or maintenance work. It would neither be necessary to know all equipment related to a fault when only substation information is required.

This issue was solved so that instead of using the *Substation* class to describe information related to substations the *Equipment* class is used instead. This is described in Figure 38.

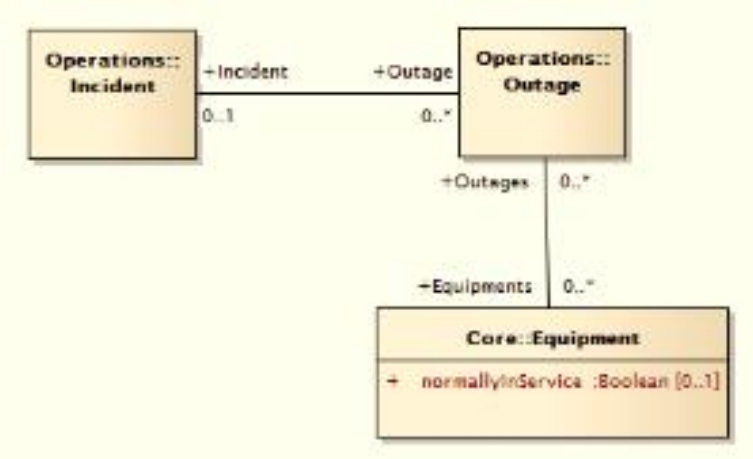


Figure 38 Substation information presented in the profile

Another option would have been to add direct association from the *Substation* class to the *Outage* class. However, in this case using the *Equipment* class was preferred, because it readily contains parameter *normallyInService* that can be used to clarify if electricity supply to the substation is normal or disrupted. This is required functionality and the *Substation* class does not have a parameter to describe the state of the substation in the CIM model and it would have to be added there. The *Equipment* class is not used for any other purpose in the profile and by using it to describe substations fewer alterations to the profile had to be made.

8 INTERFACE IMPLEMENTATION PROCESS

This chapter introduces different methods for creating the service contract for the integration of the DMS and the WMS. Testing and maintaining the system integration are both considered as important criteria when choosing a method for creating the contract.

The first three methods introduced in the subsections of this chapter are contract-first methods and the fourth method is bottom-up method. The bottom-up method means that the service is created first and the service contract is generated based on the service. The contract-first method is the exact opposite approach to the bottom-up method. In contract-first method the service contract is created first and the code skeleton is generated based on the contract. The contract-first approach gives more control over the web service, because the implementation of the service and the client is based on the service contract and not vice versa.

Subsections 8.1 and 8.2 are based on the part 100 of the IEC 61968 standard and the two other methods introduce common practices to create service contracts for web services. Common to all methods is that they require the CIM profile to be presented in a XML Schema format. The XSD file is generated from the class diagram presented in chapter 5.

8.1 IEC CIM: Generic WSDL

The use of generic WSDL is introduced in the IEC 61968 standard in part 100 to create service contracts for web services. This implementation method is based on the use of XML any-element. The purpose of the any element is to enable extensions to content models by defining a specific namespace [42].

The Generic WSDL implementation requires using Generic.wsdl and Message.xsd files provided by the IEC 61968–100 standard. Figure 39 describes the relation between the files used for the generic WSDL implementation.

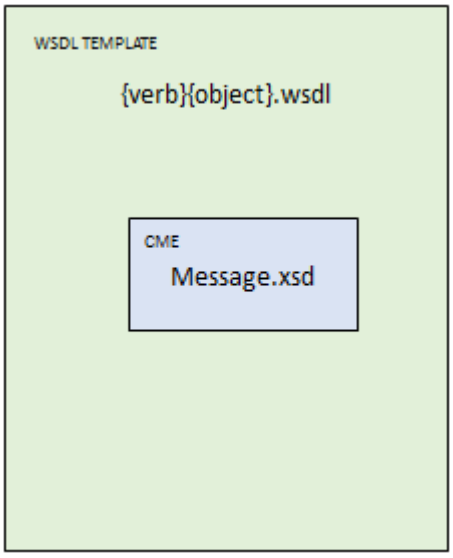


Figure 39 File structure of the Generic WSDL

The WSDL-file includes the Message.xsd file, which describes the structure of the type-independent common message envelope (CME) [39]. The generic WSDL provides three operations - *Request*, *Response* and *PublishEvent* to be used in message transferring.

Figure 40 in turn describes the payload structure for generic message container. The message content is transferred in the any-element and the *Noun* of the *Header*-element is used to describe the payload.

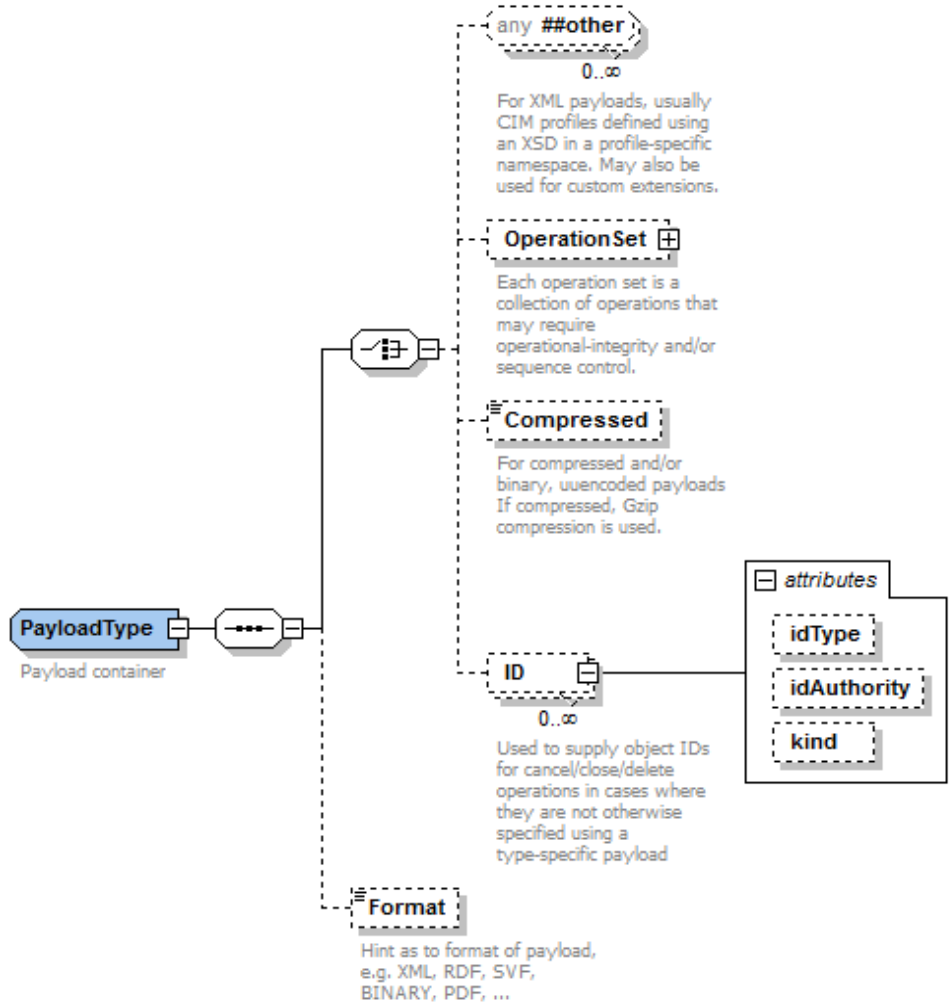


Figure 40 Generic Message Payload Container [39]

The generic WSDL enables transferring any type of XML documents and the same Message.xsd file is valid to be used with all use cases. The contract is clearly divided into an abstract level and more detailed level of information. The Generic.wsdl and the Message.xsd form the abstract level and local files form the detailed level describing the used data model. When the service contract has to be updated, the abstract level of the interface remains the same and only the local files describing the data model are updated. This is possible when the service provider and service consumers are directly in contact with each other and can agree upon changes in data model. In other cases, service consumers cannot be certain about the data model description, because the service contract does not describe the data model used.

The downside of this approach is that the data model is separate from the message. Therefore the correctness of the message content against the data model has to be checked for each received message manually. Many programming languages contain tools for XML generation and degeneration and that part of the code handling can be done automatically. If the data model would be included to the message, also the correctness could be checked automatically when receiving the message. Then, if the message content would not correspond to the data model, the message handling could be

stopped and fault message sent as a response. Now the message handling can proceed for very long before an incompatible data form appears. This is inefficient and impairs the performance of the interface and also increases the human error as the correctness of the message content is checked manually.

From the testing point of view this approach is not preferred either, because when the data model is separate from the message it requires more manual work than when the data model is included in the message.

8.2 IEC CIM: Strongly-typed WSDL

Strongly-typed WSDL is another implementation method provided by the IEC 61968-100 standard. The standard provides three wsdl templates and two type-specific message templates. The right template is chosen according to the request type. WSDL template 1 is used with *Get* request, template 2 with *Send*, *Receive* and *Reply* requests and template 3 with *Request* and *Execute* type of requests. Accordingly, message template 1 is used with *Send*, *Receive*, *Reply*, *Request* and *Execute* requests and message template 2 with *Get* and *Reply* requests. [39]

Figure 41 describes the file structure for the strongly-typed WSDL implementation. The WSDL file includes the type-specific message file, which in turn imports the file containing the common message envelope (CME) and the XML Schema file describing the profile. The interface description is then created based on these files.

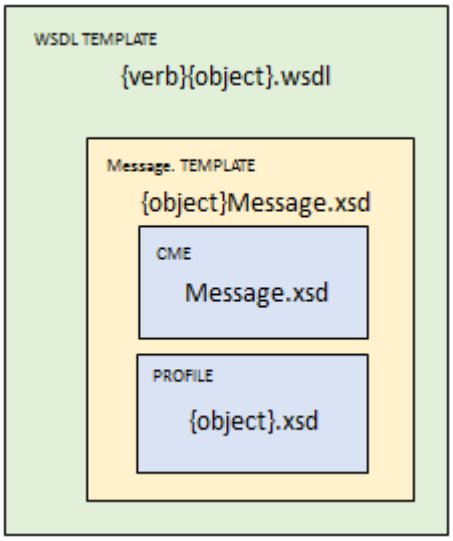


Figure 41 File structure of the strongly-typed WSDL

Figure 42 describes how the payload in the type-specific WSDL implementation is placed. The CME structure marked with blue as *Common* comes from the Message.xsd

and unlike with the Generic WSDL, the profile is actually included in the payload. This is illustrated in the *Type-Specific Payload* structure marked with orange.

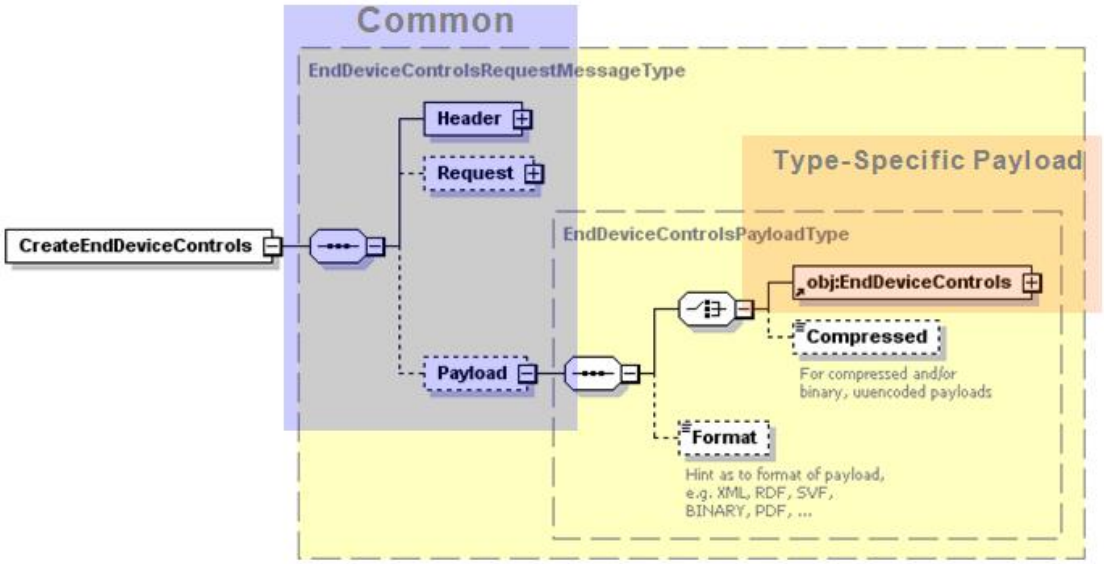


Figure 42 Type Specific Message Payload Container [39]

Constructing the service contract includes several steps and for that reason creating the contract with this method may seem complicated. Using the contract on the other hand is simple because it is self-descriptive and can be validated automatically. Also automated testing of the contract is possible because the data model is included in the message.

If the same service contract is used to complete several use cases, the size of the profile expands easily. The contract may become hard to read, but this downside can be compensated with comprehensive documentation.

8.3 Generating WSDL

Generating WSDL first is also a contract-first type method, but unlike in the first two methods presented, here WSDL document is created from scratch. In this thesis, Enterprise Architect was applied to model WSDL document, but WSDL generation could also be done without such tool. However, writing service contract from scratch without any automatic tools is found to be inefficient and error-prone. The EA uses UML visualization techniques that help to understand the development process. According to WSDL structure, EA generates 5 packages called *Types*, *Messages*, *PortTypes*, *Bindings* and *Services*. Package called *Types*, contains the information of different data types transmitted. *Messages* package defines input and output messages. In *PortTypes*, sup-

ported operations are defined. *Bindings* package contains the information how messages are transmitted and *Services* package describes where each service is located. Figure 43 illustrates WSDL generation with EA.

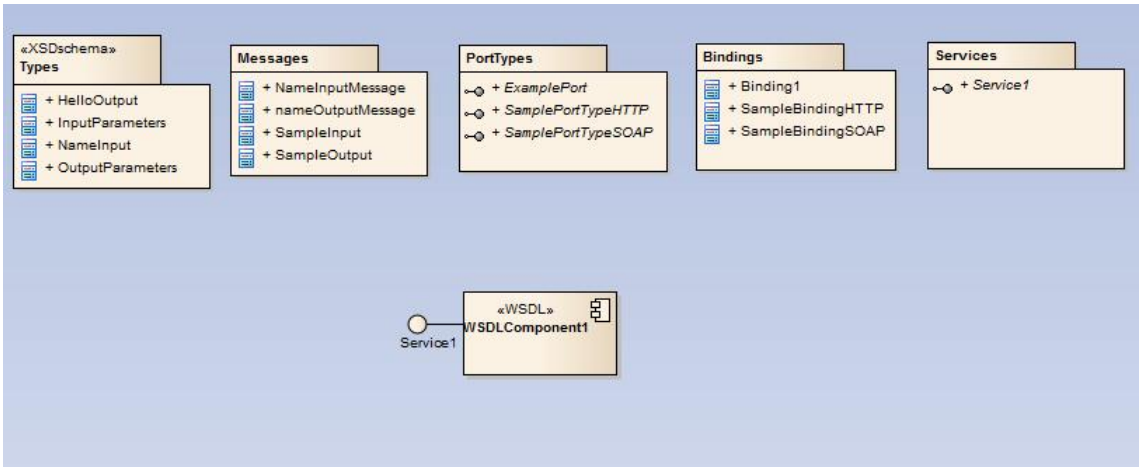


Figure 43 Generating WSDL document with EA

The benefit of this method is that the WSDL document structure can be designed to correspond to the current requirements. This means that all extra operations and message structures are left out of the WSDL document. Thus, reading and understanding the service contract is thus easier as only the required functions and message structures are defined in the WSDL document.

With this tool, it is possible to create several services and later decide which ones are exposed to the interface. This is useful for example in a situation when several version of the same service exists.

In addition to WSDL generation, the EA also supports importing WSDL documents and modifying them in EA. This functionality enables reusing existing WSDL documents if only few modifications are required. Sparx Systems provides more information how to use EA to create WSDL documents at [43].

8.4 Service First

Generating the service first is a bottom-up method for creating web services. In this method the service is coded first and the interface for the service is generated based on the code. This is commonly used approach in web services programming, because it is fast and easy way to create web services and does not require any knowledge of XSD nor WSDL syntaxes. There are also multiple programming tools available for this implementation approach.

This bottom-up approach, however, is found to be inconvenient what comes to maintaining the service and abstracting the interface. This approach generates services

that are very developer centric, because the interface is based on the service application. The danger of this approach is that the interface of this type of service does not address the needs of service consumers. Abstracting the interface is also harder with this method in comparison with the contract-first method. Changes in the internal implementation of the service affect the service contract and therefore also the client applications. This is not desirable quality for a web services, because the aim is to keep the contract unchanged for as long as possible.

Service first method is sometimes used to generate a WSDL document, which is later used as basis for contract-first implementation method. This combines benefits from both approaches to create web services. The bottom-up method is used to easily create WSDL document and the contract-first method ensures better maintenance qualities for the service contract.

9 RESULTS

Information system integration consists of different steps including requirements capture, designing the interface, implementing the interface and maintaining and testing the system integration. Requirements capture was discussed in detail in Chapter 5 and use cases defined for this system integration were introduced in Chapter 6. This chapter explains the used methods and techniques when integrating the DMS and the WMS.

9.1 Use of Standards

This thesis studied the use of standards in integration of the DMS and the WMS. Using standards in the integration was not an intrinsic value. Instead, the purpose was to evaluate the suitability of the CIM to function as a canonical data model in the integration of the DMS and the WMS. The use of standards was not expected to provide a plug-and-play functionality for the systems, but it was expected to ease the integration process in comparison to not applying any standards in the integration. At the time of writing this thesis, the CIM standard was still under development and some parts of the CIM had no more than a draft of the standard published.

The current version of the CIM standard provided readymade profiles for different business functions to be used for both internal and external data handling in information systems, but lacked a profile for exchanging work management information. A profile for transferring data between the DMS and the WMS was constructed in this thesis. The CIM standard was used as a basis in creating the profile for data exchange between the two systems. A profile purely based on the CIM was found not to satisfy the requirements set for the system integration and some alterations to the standard model were made. The profile constructed in this thesis is described in detail in chapter 5.

Even though the profile could not be constructed purely based on the CIM standard, using the CIM standard was found to benefit the system integration otherwise. Multiple data structures were adapted directly from the CIM standard to the profile, which reduced the amount of work in the designing phase. Using these readymade data structures probably reduced the time spent on designing and improving the profile as most of the class relations and multiplicities were adapted directly from the standard.

Instead of using the CIM standard as a basis in constructing the profile, an alternative approach would have been to construct the profile from scratch. A benefit of this

approach would have been that the profile could have contained fewer classes, which would have made the profile simpler. On the other hand, this would have created a whole new data model. Designing a new data model and testing its suitability would have required more effort than using the readymade structures.

In addition to the CIM standard also the use of the XML and its extensions and the WSDL in the integration can be considered as using standards in this system integration. XML and WSDL are both standard ways to present information in web services. The use of the XML and the WSDL enabled automatic code generation from the service contract. The use of tools to automatically generate the code made the development process faster and helped to minimize human error in coding phase. The best practice found was to use the CIM model as a basis and to do all alterations to the model at the profile level. When the changes were made in the class diagram presentation of the profile, software tools could be used to automatically generate the code.

9.2 Integration Architecture

Different integration architectures were discussed in chapter 3. In integrating the DMS and the WMS the point-to-point integration architecture was found to be the most suitable integration architecture. The point-to-point integration was chosen because it is easy and straightforward to implement in this kind of system environment. The main weakness of the point-to-point integration architecture is the scalability, which in this case is not an issue as only two systems are integrated with each other.

Other integration architectures introduced were the EAI and the ESB. The benefits of the EAI and the ESB can be seen in system environments that are more complex and require better scalability than what is the case with the DMS-WMS integration. Integrating the DMS and the WMS with the EAI or the ESB would not be justified, because these integration architecture styles would just remarkably increase the complexity and expenses without truly benefiting the integration. As long as the complexity of the system environment from the point of view of the DMS and the WMS integration does not remarkably increase and require better scalability, the point-to-point integration is the recommended integration architecture.

9.3 System Environment

The DMS and the WMS are required to be able to communicate with each other over the Internet, because these two systems are not necessarily located within the same company. To enable the communication over the Internet, web services technology is used. Web services provide a platform independent communication technology, which is preferable feature as the integrated systems are produced by different vendors that may use different platforms for development.

Bidirectional communication between the DMS and the WMS is required so that information can be synchronized between the systems. Different techniques to establish bidirectional communication between the systems were discussed in Chapter 3.5. Communication techniques over the web can be categorized into push and pull techniques. Traditional web architecture is designed to support the pull technique where the client sends a request to the server and the server processes the request and responds to the client. The traditional server-client technique is also used in the integration of the DMS and the WMS.

In order to establish two-way communication between the systems two sets of client-service pairs were required as presented in Figure 44. The DMS provides a service to receive the updates from the WMS as presented in the upper client-service pair in the Figure 44. The lower pair transfers the DMS updates to the WMS. To enable this connection the firewalls must be opened. Also the response times of the client and the service must be set to be long enough so that the client waits until the service has processed the request and formed a response to the client. The traditional server-client technique is widely used with point-to-point integration architecture and it was familiar technique to developers.

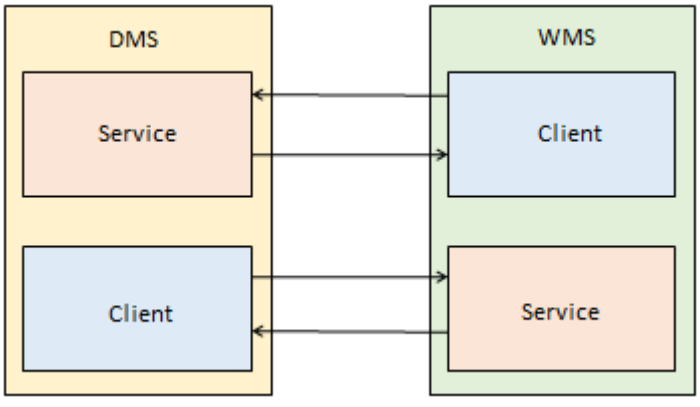


Figure 44 The DMS –WMS communication model

The other techniques were unfamiliar to developers and as no additional benefit of using them was seen they were discarded. The main issue with the polling technique was setting the polling frequency, because the events in this system occur with different fre-

quencies. Setting a long polling interval would have resulted in weaker accuracy and decreased the freshness of data. Shorter polling interval on the other hand would have increased the server load even in a case when no events truly occur. With the long polling technique the polling frequency problem would have been removed, but using the long polling requires creating two clients; one to poll changes from the service side and another to send updates from the client side. In the end, long polling set up would have not be so different from the more familiar technique of using two sets of client-service pairs. The server push technique uses the publish/subscribe model to send information to the client, but in order to transfer data from the client to the server the more traditional request/response model is also required. Again, using familiar client-server was preferred.

The system environment was designed so that it fulfills the security requirements stated for this system integration. From the three aspects of information security introduced in Chapter 5.2 data integrity was found to be the most significant. The other two aspects of information security, confidentiality and availability, were classified as level low in this system integration. Ensuring data integrity is important because violation of data integrity can in worst case cause longer outages and therefore has effect on compensation payables. In order to ensure that the data does not change on the way when transferred between the two systems and that the data is received from a trusted source several methods can be used. For example, using HTTPS instead of HTTP provides secure communication between the two systems and certificates can be used to confirm the source of received data. Technology decisions made for integrating the DMS and the WMS do not restrict the use of security methods. Thus, required level of information security can be ensured for this system integration.

9.4 Implementation of the Interface

When juxtaposing different implementation methods covered in chapter 8 it can be seen that they all have their strengths and weaknesses. In development of web services using contract first method is commonly preferred over the bottom-up method, because contract-first method gives more control over the web service. Three out of four methods introduced in chapter 8 were contract first methods. From the contract first methods, the strongly-typed WSDL method based on the part 100 of the IEC 61968 standard was used in integrating the DMS and the WMS.

The strongly-typed WSDL method was chosen over the other two contract first methods, because it was fast to implement and easy to maintain and test. In comparison to the other implementation method provided by the IEC 61968-100 called Generic WSDL, combining the templates provided by the standard is a bit more complicated with this chosen method. However, when the contract is created, maintaining and test-

ing proved to be easier with the strongly-typed WSDL than with the Generic WSDL. The service contract created based on the strongly-typed WSDL can be validated automatically, because the data model is included in the message. Regardless of the fact that the interface implementation is done purely according to the CIM standard, the content of the contract is not restricted to the CIM. The interface content is determined by the profile and as long as the XML Schema generated from the profile is valid according to the XML standard, the interface can contain anything stated in the XML Schema file.

In the third contract-first method introduced, the service contract was created from the scratch. The obvious benefit of this method is that when the service contract is custom made, only the required operations and message structures are included in the contract. This makes the contract easier to read and adapt, but creating it is time consuming and prone to errors.

The communication technique chosen for the service-client has an effect on which WSDL template and Message template are to be used. When using the strongly-typed WSDL, the WSDL template 3 and the Message template 2 were chosen so that the service contract would support the *Request/Response* message structure required for the traditional service-client communication. If the Server push technique would have been chosen to be the communication technique, a WSDL and Message templates would have to be chosen so that they had supported *Publish/Subscribe* message structure. Both the WSDL and the Message templates are provided in the IEC 61968-100 standard.

9.5 Integration Tools

System integration process requires using tools to ease the process and to minimize the impact of design changes. The use of tools also enables automatic code generation, minimizes the need for writing documents and files by hand and helps in testing.

Design changes occurred when the interface needed to be updated. This required regenerating the code, which was easier when it could have been done automatically. Minimizing the code alteration by hand was preferred, because it is more prone to errors than code generated automatically. Also regenerating and maintaining the interface and documenting the integration process were easier when manual alteration of the code was minimized.

Available tools were examined in this thesis in order to find suitable tools for the integration process. Software tools and their use in this thesis are introduced under the following subsections.

9.5.1 Visual Studio

Visual Studio is an Integrated Development Environment (IDE) by Microsoft. It provides a platform for software development in different programming languages including C#, C++ and Java and enables development of web sites, web applications, web services etcetera.

Implementation of the interface for the DMS side service and client was completed using Visual Studio 2012.

9.5.2 Enterprise Architect and CIMEA

The Enterprise Architect (EA) is a modeling and designing tool by Sparx Systems.
The EA supports multiple standards including the UML, the WSDL and the XSD and provides a platform for visual modeling. Developing UML class diagrams and sequence diagrams with this tool is fast and easy to learn. The EA requires purchasing a license.

The CIMEA is an add-in for the EA that enables generation of CIM profiles. All classes, class relations and parameters included in the CIM model are imported to the EA with this add-in and CIM compatible class diagrams can be generated.
The CIMEA provides a free tool to view the CIM model, but in order to create profiles and alter the model a license is required. The CIMEA was used in this thesis to build the profile and to generate XML Schema for the interface.

There are multiple other CIM related tools that can be found as freeware. Other available tools for CIM handling include InterPSS OpenCIM, jCleanCIM, CIMvian, CimContextor, CIMphony, MD3i, IBM Rational Rose, CIMTool and CIMSpy.

9.5.3 SOAPUI

The SOAPUI by SmartBear is a testing solution with a graphical user interface. It provides support to test web services created for example with .NET, J2EE, Perl or PHP. The only requirements are that the tested web service has a defining WSDL, it uses SOAP/HTTP binding and does not use SOAP-Encoding. [44]

In this thesis SOAPUI was used to do the module testing for the DMS.

9.6 Testing

This subsection discusses the testing mainly from the DMS point of view as the two systems were tested partly separate from each other. Overall testing requirements for the system integration of the DMS and the WMS included module testing, integration testing and user testing.

Module testing for the DMS was done using a test client and a test service. Later a program called SOAPUI was used instead of test client. Creating and changing information in a test client and in a test service was a little more time consuming than using the SOAPUI. Therefore the remaining module testing was done with this software tool. Module testing revealed mistakes done in mapping the data from the interface to the internal data structure of the DMS and vice versa. Module testing was required, because the two systems were produced by different vendors and tested separately. Testing the systems first separately probably made the next testing phase quicker because myriad of errors were detected and then corrected already at this phase. However, this claim cannot be proved, as module testing was done to all functions in the interface.

After module testing had been done for both systems integration testing was carried out. When problems and mismappings of the data between the systems occurred, mistakes were fixed and module testing repeated for those parts. Integration testing ensured that the connection between the two systems can be established and that data flows from one system to the other as defined. As this phase of testing was completed successfully the next step was user testing.

In user testing, it was preferred that the actual end users of both systems participated in testing. User testing of the DMS-WMS integration was done in a DSO by personnel who regularly operate either of the two systems. This way the testing was as close to the real operating situation as possible. User tests revealed problems in user interface level if the data mappings were erroneous.

Testing of this system integration ended up being an iterative process. When mistakes were found in any part of testing, errors were fixed and then testing was done again starting from module testing. Only in very simple and straightforward cases, some parts of the testing were skipped in order to save time. Integration testing and user testing were carried out in a testing environment, which was a copy of the production environment. The system integration was ready to be used in a production environment when required functionality for the integration of the DMS and the WMS was achieved without mistakes.

10 CONCLUSION

This thesis studied the integration of a distribution management system and a work management system using standard interfaces. All steps of the information system integration were studied and evaluated in this work. First, requirements for this system integration were captured and then an interface was designed to fulfill these requirements. Different integration architectures, communication technologies and methods for constructing a service contract were evaluated. Finally, a service contract was built according to the strongly-typed WSDL method provided by the IEC 61968 standard and the system integration was implemented based on the contract. The concrete result of the thesis was an integration of the ABB MicroSCADA Pro DMS600 distribution management system and a work management system.

The use of standards in information system integration was expected to ease the integration process and decrease the maintenance costs. The IEC CIM standard studied in this thesis was found to be still too immature to be directly used in constructing the interface for this system integration. The CIM standard is still under development and at the time of writing, the standard still lacked profiles for transferring data in some business functions. A profile to transfer data related to work management was constructed in this thesis. The IEC CIM standard was used as a basis in the profile, but in order to fulfill the requirements defined for the system integration some alterations to the standard data model were made. Even though the profile required some alterations to the standard, the service contract was implemented according to the CIM standard. The strongly-typed WSDL method provided in the part 100 of the IEC 61968 standard proved to be suitable method for implementing a service contract for this system integration. A service contract constructed according to this method was easy to maintain and test.

In future, the profile can also be updated to correspond to the CIM standard if necessary. This change will be fairly easy as the majority of the current profile is already done according to the standard.

The integration of the DMS and the WMS can be used to improve the communication between control center personnel and crews operating in the field. This improves the DSO’s ability to prevent and manage outages in the electricity supply, which increases the reliability of electricity distribution. In addition, this system integration can also be used to improve the customer service during outages.

REFERENCES

[1] Työ- ja Elinkeinoministeriö, "Työ- ja elinkeinoministeriön ehdotus toimenpiteistä sähkönjakelun varmuuden parantamiseksi sekä sähkökatkojen vaikutusten lievittämiseksi," 16.3.2012.

[2] A. Koto, Tietojärjestelmien väliset rajapinnat sähkönjakeluverkon käyttötoiminnassa. Master of Science Thesis, Tampere: Tampere University of Technology, 2010, p. 102.

[3] T. Väre, Verkkotietojärjestelmiin perustuvan palveluliiketoiminnan kehittäminen. Master of Science Thesis, Tampere: Tampere University of Technology, 2007, p. 99 + app. 3p (in Finnish).

[4] A. Aminoff, I. Lappeteläinen, J. Partanen, S. Viljainen, K. Tahvanainen, P. Järventausta and P. Trygg, "Ostopalveluiden käyttö verkkoliiketoiminnassa [Outsourcing services in electricity distirbution network industry]," VTT Tiedotteita - Research Notes 2462, Espoo, 2009.

[5] J. Toivonen, P. Trygg, S. Antila, A. Mäkinen, P. Järventausta, P. Mäenpää, V. Nyrhilä, H. Saaristo and J. Mattsson, Sähköyhtiöiden tietojärjestelmäkartoitus., Tampere & Vaasa: Tampere University of Technology, Department of Power Engineering, 2005, p. 40 + app. 16 p.

[6] R.-M. Keski-Keturi, Implementing the IEC Common Information Model for Distribution System Operators. Master of Science Thesis, Tampere: Tampere University of Technology, 2011, p. 91 p. + app 2p..

[7] E. Lakervi and J. Partanen, Sähkönjakelutekniikka, Helsinki: Otatieto Helsinki University Press, 2009.

[8] P. Verho, "Configuration Management of Medium Voltage Electricity Distribution Network," Tampereen teknillinen korkeakoulu, Tampere, 1997.

[9] V. Hälvä, Development of Process Data Utilization in Proactive Network Management. Master of Science Thesis, Tampere: Tampere University of Technology, 2012, p. 84.

[10] P. Verho, P. Järventausta, M. Kärenlampi, H. Paulasaari and J. Partanen, Distribution management system - Overall system description, Tampere: Tampere University of Technology Power Engineering Report 1-96, 1996.

[11] G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, Addison-Wesley, 2003.

[12] R. Eerola, Analysing Integration and Information Security: Enterprise Service Bus Solution for Smart Grid. Master Thesis, Tampere: Tampere University of Technology, 2013, p. 112 + app. 4 p.

[13] D. Becker, H. Falk, J. Gillerman, s. Mauser, R. Podmore and L. Scheneberger,

Standards-based approach integrates utility applications, vol. 13, IEEE Computer Applications in Power, 2000, pp. 13-20.

[14] M. Rix, "Bottom Line SOA: The Economics of Agility," 23 April 2007. [Online]. Available: http://www.marcrix.com/resources/BottomLineSOA.pdf. [Accessed 12 December 2013].

[15] Electric Power Research Institute, "The Common Information Model for Distribution – An Introduction to the CIM for Integrating Distribution Applications and Systems," *Technical Update 1016058,* p. 98, 2008.

[16] F. Curbera, T. J. Watson, M. Duftler, R. Khalaf and W. Nagy, "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI," *Internet Computing, IEEE,* vol. 6, no. 2, pp. 86-93, 2002.

[17] T. Erl, Introduction to Web Services Technologies: SOA, SOAP, WSDL and UDDI, Prentice Hall, 2004.

[18] d. Sprott and L. Wilkes, "Understanding Service-Oriented Architecture," *Architecture Journal,* 2004.

[19] P. A. Laplante, J. Zhang and J. Voas, "What's in a Name? Distinguishing between Saas and SOA," *IT Professional*, vol. 10, no. 3, pp. 46-50, June 2008.

[20] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris and D. Orchard, "Web Services Architecture," 11 February 2004. [Online]. Available: http://www.w3.org/TR/ws-arch/. [Accessed 20 October 2013].

[21] T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maler and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)," 26 November 2008. [Online]. Available: http://www.w3.org/TR/REC-xml/. [Accessed 6 October 2013].

[22] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H. F. Nielsen, A. Karmarkar and Y. Lafon, "SOAP Version 1.2 Part2 : Adjuncts (Second Edition)," 27 April 2007. [Online]. Available: http://www.w3.org/TR/2007/REC-soap12-part2-20070427/. [Accessed 31 July 2013].

[23] IEC, "Implementation Profiles for IEC 61968," 2012.

[24] E. Bozdag, A. Mesbah and A. van Deursen, *A Comparison of Push and Pull Techniques for AJAX,* Paris: IEEE, 2007, pp. 15-22.

[25] IBM, "Publishing and subscribing to server events from an Ajax client," [Online]. Available: http://publib.boulder.ibm.com/infocenter/wsdoc400/v6r0/index.jsp?topic=/com.ibm.websphere.ajax.devguide.help/docs/PureAjax_pubsub_clients.html. [Accessed 12 November 2013].

[26] X.-H. Sun, S. Byna and Y. Chen, "Improving Data Access Performance with Server Push Architecture," *Parallel and Distributed Processing Symposium,* pp. 1-6, 26-30 March 2007.

[27] G. Robinson and M. Zhou, "Utility applications should be integrated with an

interface based on a canonical data model, not directly with each other," in *Proceedings of the IEEE Power Systems Conference and Exposition (PSCE)*, New York, 2004.

[28] G. McNaughton and B. Saint, "Harmonization of the Common Information Model and Multispeak," Phoenix, AZ, 20-23 March 2011.

[29] Electric Power Researh Institute, "Common Information Model Primer," [Online]. Available: http://www.epri.com/abstracts/Pages/ProductAbstract.aspx?ProductId=000000000001024449. [Accessed 25 July 2013].

[30] "CIM Users Group," UCA Internation Users Group, [Online]. Available: http://cimug.ucaiug.org/default.aspx. [Accessed 8 October 2013].

[31] I. E. Commission, "TC 57 Power systems management and associated information exchange," 2013. [Online]. Available: http://www.iec.ch/dyn/www/f?p=103:23:0::::FSP_ORG_ID,FSP_LANG_ID:1273,25. [Accessed 28 October 2013].

[32] International Electrotechnical Commission, "Application integration at electric utilities - System interfaces fors distribution management - Part 1: Interface architecture and general recommendations," 2012.

[33] International Electrotechnical Commission, "Application integration at electric utilities - System interfaces for distribution management - Part 100: Implementation profiles," 2013.

[34] N. R. E. C. Association, "MultiSpeak," 6 May 2013. [Online]. Available: http://www.multispeak.org/Pages/default.aspx. [Accessed 15 October 2013].

[35] J. J. Simmins, "Common Information Model (CIM) and MultiSpeak for Smart Grid," in *Smart Grid demonstration Advisory Meeting*, Clamart France, 2010.

[36] H. van Vliet, Software Engineering: Principles and Practice, 3 ed., West Sussex: John Wiley & Sons Ltd, 2008.

[37] B. Nuseibeh and S. Easterbrook, "Requirements Engineering: A Roadmap," New York, USA, 2000.

[38] I. Nikander, "Cleen portal," 23 August 2008. [Online]. Available: https://portal.cleen.fi/sgem/work_packages/wp6 [restricted access]. [Accessed 11 November 2013].

[39] Energiateollisuus ry, *Keskeytystilasto-ohje 2012 v 4.0*, 2012.

[40] J. Weiss, "Assuring Industrial Control System (ICS) Cyber Security," 25 August 2008. [Online]. Available: http://csis.org/files/media/csis/pubs/080825_cyber.pdf. [Accessed 7 November 2013].

[41] M. Pekkarinen, "Sähkönkulutuksen mittauksen uudistus," Slide show at press conference, 5 February 2009. [Online]. Available: http://www.tem.fi/files/21782/sahkonkulutuksen_mittauksen_uudistus050209.pdf

. [Accessed 17 July 2013].

[42] D. C. Fallside and P. Walmsley, "XML Schema Part 0: Primer Second Edition - W3C Recommendation," 2004.

[43] Sparx Systems, "Create WSDL 1.1 Model Structure," [Online]. Available: http://www.sparxsystems.com/enterprise_architect_user_guide/9.2/soa_and_xml/create_wsdl11_model_structure.html. [Accessed 15 November 2013].

[44] SmartBear, ”SoapUI,” 2013. [Online]. Available: http://www.soapui.org/. [Haettu 3 February 2014].

[45] L. Nordström, ”Use of the CIM Framework for Data Management in Maintenance of Electricity Distribution Networks,” Stockholm, 2006.

[46] A. W. McMorran, An Introduction to IEC 61970-301 & 61968-11: The Common Information Model, Glasgow: Institute for Energy and Environment, Departmetn of Electronic and Electrical Engineering, 2007, p. 42.

[47] ”IEEE Guide for Electric Power Distribution Reliability Indices,” IEEE Power & Energy Society, New York, 2012.

[48] J. Chetty and M. Coetzee, "Towards an information security framework for service-oriented architecture," *Information Security for South Africa (ISSA), 2010*, pp. 1-8, 2-4 August 2010.

[49] J. Ekanayake, K. Liyanage, J. Wu, A. Yokoyama and N. Jenkins, Smart Grid: Technology and Applications, John Wiley & Sons, Ltd, 2012.

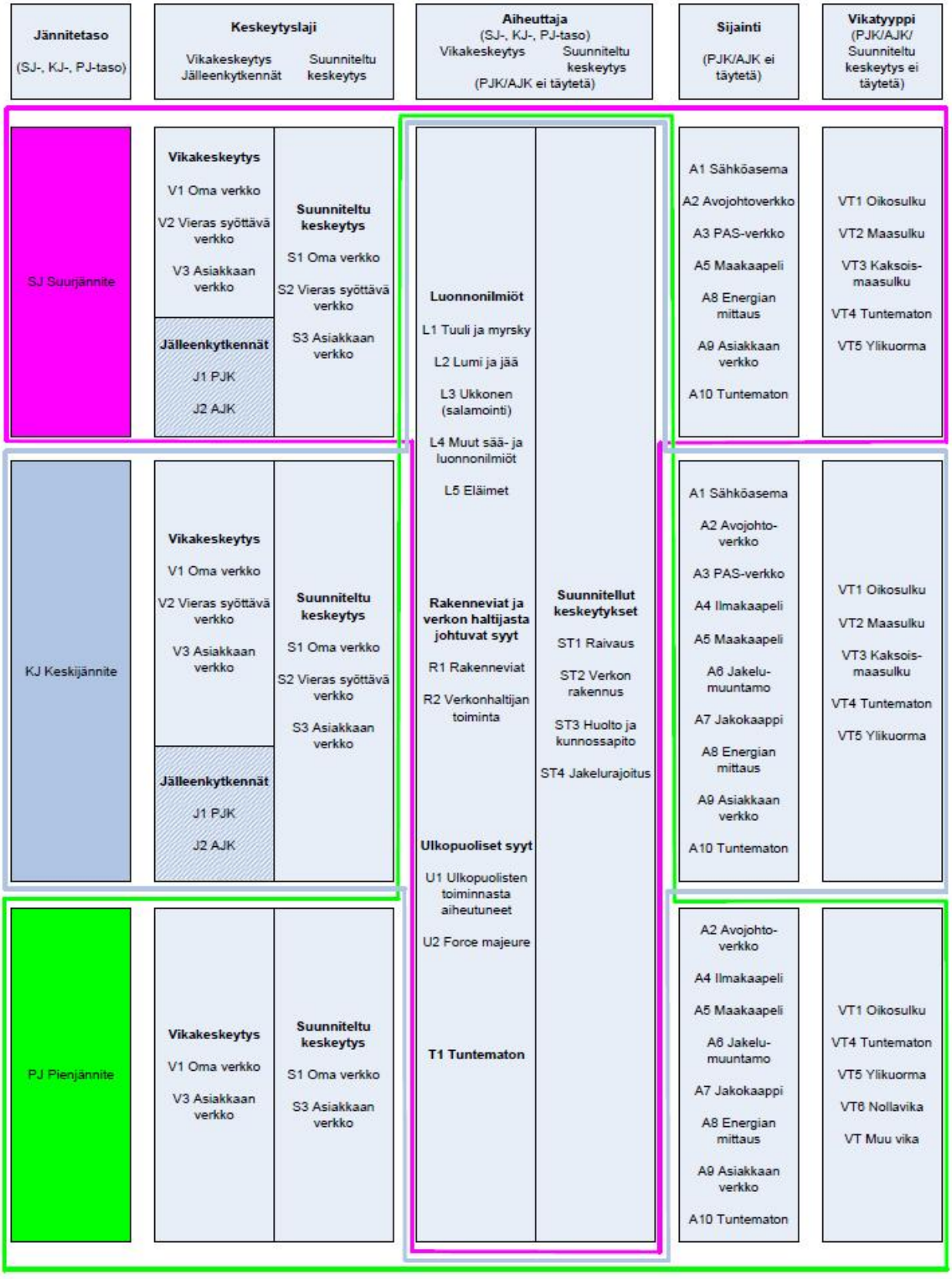
[50] Finnish Energy Industries, "Finnish Energy Industries," 2013. [Online]. Available: http://energia.fi/en. [Accessed 13 November 2013].

[51] P. Järventausta, S. Repo, A. Rautiainen and J. Partanen, "Smart grid power system control in distributed generation environment," *Annual Reviews in Control*, vol. 34, no. 2, pp. 277-286, 2010.

[52] G. A. McNaughton, G. Robinson and G. R. Gray, "MultiSpeak® and IEC 61968 CIM: Moving Towards Interoperability," in *Proceedings of the Grid-Interop Forum*, Atlanta, USA, 2008.

APPENDIX A

Outage Reporting Requirements for Finnish Energy Industries (ET).



APPENDIX B

The profile used for the interface to integrate the MicroSCADA Pro DMS600 by ABB Ltd and the WMS.